EXAMENSARBETE INOM DATATEKNIK,
GRUNDNIVÅ, 15 HP

STOCKHOLM, *SVERIGE 2016*

Evaluation of Recommender System
Utvärdering av rekommendationssystem

CHRISTOFER DING

KTH
SKOLAN FÖR TEKNIK OCH HÄLSA

Evaluation of Recommender System

Utvärdering av rekommendations-system

Christofer Ding

Degree project in
Computer Engineering,
First cycle, 15 credits
Supervisor at KTH: Reine Bergström
Examiner: Ibrahim Orhan
TRITA-STH 2016:51

KTH
School of Technology and Health
136 40 Handen, Sweden

Sammanfattning

Rekommendationssystem har blivit en av de viktigaste beståndsdelar för många företag, såsom YouTube och Amazon. Ett rekommendationssystem består av en serie av algoritmer som förutsäger och rekommenderar produkter till användare. Denna rapport omfattar valet av många öppen källkod rekommendationssystem projekt, och filmprognoser görs med det valda rekommendationssystemet. Baserat på filmprognoser, gjordes en jämförelse mellan *precision* och en *förbättrad precision* algoritmer.

Det valda rekommendationssystemet använder singulärvärdesuppdelning som kollaborativ filtrering. Baserat på rekommendationsresultat som produceras av rekommendationssystemet, jämförelsen mellan *precision* och den *förbättrade precisions* algoritmer visade att resultatet av *förbättrad precision* är något högre än *precision* i olika brytvärden och olika dimensioner av egenvärden.

Nyckelord
Rekommendationssystem, singulärvärdesuppdelning, precision, rekommendations noggrannhet

Abstract

Recommender System (RS) has become one of the most important component for many companies, such as YouTube and Amazon. A recommender system consists of a series of algorithms which predict and recommend products to users. This report covers the selection of many open source recommender system projects, and movie predictions are made using the selected recommender system. Based on the predictions, a comparison was made between *precision* and an *improved precision* algorithm.

The selected RS uses singular value decomposition in the field of collaborative filtering. Based on the recommendation results produced by the RS, the comparison between *precision* and the *improved precision* algorithms showed that the result of *improved precision* is slightly higher than precision in different cutoff values and different dimensions of eigenvalues.

Keywords
Recommender system, singular value decomposition, *precision*, recommendation accuracy

Preface

This report covers a thesis work in the third year in computer engineering, program and system development at School of Technology and Health, KTH. The goal of this thesis work is to create a prototype of recommendation system for PlayPilot.

The report is written assuming the readers have the fundamental knowledge of statistics, linear algebra and programming concepts.

I would like to thank PlayPilot to give me the opportunity to do this fascinating project. My supervisors at PlayPilot, Emil Wikström and Gustaf Sjöberg, the CEO of PlayPilot Adam Chrigström. In addition, I would like to thank my supervisor Reine Bergström at KTH for his patience and guide me to reach the goal using scientific methods.

Table of contents

1 Introduction

Today, Recommender Systems (RS) are ubiquitous on the Internet. RS make recommendations based on data related to users and products. For instance, the RS on YouTube would recommend related videos to its users. Because users have different requirements and preferences, the recommendations must be personalized.

1.1 Problem statement

Each website, as a service, is always expecting that users rely on their service as much as possible. For websites containing large quantity of products, only a small portion can be displayed. Therefore, these services need a system to make recommendations. The purpose of RS is to predict and display user's favorite products with minimal number of clicks. The products to be recommended could be news, jobs, music, movies and peoples. Since the users have different preferences, the recommendation to each user must be personalized.

A Swedish company called PlayPilot requires a RS to improve its user experience. PlayPilot continuously collects streaming resources from different websites and presents them on their website with an external link. These resources include movies, TV series, documentaries and other TV programs.

1.2 Goals

The primary goal of this thesis work is to select an open source recommender system based on which kind of data collected by PlayPilot and evaluate the recommendation accuracy using different algorithms.

Research of recommender system

- To research the advantages and disadvantages of various kinds of filtering approaches of recommender system
- To describe and analyze challenges in recommender systems

Research and selection of existing open source recommender systems

- To research open source recommender systems and select the most suitable one for PlayPilot by considering from different aspects

Data collection and data preprocessing

- To analyze the relevant data required for different filtering methods

Training and evaluation of selected recommender system

- To train the selected recommender system by using existing user data
- To use the recommender system to make personalized movie recommendations based on preferences of each user
- To evaluate the recommendation results by using precision algorithm and improved precision algorithm

1.3 Delimitations

- The selected RS only contains a console application in order to allow the developers at PlayPilot to test and evaluate.

- The RS selected for PlayPilot might not be suitable to provide recommendations for other web applications.

- Due to the limitation of the timeline and cost of this project, the system needs not be integrated with PlayPilot's web application.

- The system needs not be deployed and tested in a real world scenario.

1.4 Author of ownership

The dataset which to train and evaluate the RS belongs to GroupLens, a research lab in the Department of Computer Science and Engineering at the University of Minnesota.

2 Theory and background

This chapter covers an overview of open source recommender system (RS), concepts, terminologies and common problems of RS, the importance of data collection and data preprocessing, different method of split data into training and test set. Lastly, how to evaluate the recommendation results using different algorithms.

2.1 Working steps of Recommender Systems

The working principle of all RS is to predict ratings for each item and each user. The prediction is based on user's earlier interactions with the system. Both viewing and purchasing a product indicates that user is showing interest about it. It means that all measurable facts, such as which items was viewed, how long does a user viewing each item by users could be converted into ratings. These methods are called implicit ratings. A simpler way would be to allow users to vote directly which is named explicit ratings. More explanation about ratings will be covered in section 2.3.2.

These ratings are the basis for personalized recommendations. These ratings are going to be saved in a file and loaded as a matrix in RS. Each row of the matrix represents a user and each column represents an item. Since users only get involved with a very small proportion of the numbers of all products, the data in matrices are extremely sparse. For example, even though a user on Amazon get involved (viewed, purchased) with a lot of products, it is still a tiny fraction of all products in Amazon. The products which users do not get involved are set to zero by default. The next step is how RS computes using different approaches which is described in section 2.2.

After a series of computation, all the zeros in the matrix will be replaced by predicted ratings. Finally, the predicted ratings for each user are sorted in descending order and only the top few items are recommended. The list which contains the top few items often referred as the "recommendation list" or "top n list", where n is a natural number greater than zero. The selection of the size of n depends on how the graphical user interface is designed.

2.2 Approaches of Recommender Systems

There are many different approaches to implement a recommender system. Content-based Filtering computes the similarity between items. On the contrary, Collaborative Filtering finds similarities between users.

2.2.1 Content-based Filtering

Content-based filtering analyzes the similarity between items. It recommends items based on the items which users have involved in the past. Content-based filtering requires detailed data of items. A set of detailed data could either be the price or be category which the item belongs to. Furthermore, if an item belongs to more than one category, the detailed data could be the weight of each category. For example, movie *Forrest Gump* was tagged by two genres: drama and romance. Each genre has its weight, 0.7 for drama and 0.3 for romance.

2.2.2 Collaborative Filtering

Collaborative filtering finds users who have similar interest or taste. It computes similarity between users [1].

2.2.2.1 Item-Based Collaborative Filtering

Item-Based Collaborative Filtering (IBCF) computes the similarity between two items. Unlike content-based filtering, comparison of features is not the property of items. It is the behavior from users.

2.2.2.2 User-Based Collaborative Filtering

User-Based Collaborative Filtering (UBCF) matches users who have similar taste or interest. It focuses on what users have done, and ignores what users have not done.

Unlike collaborative filtering, content-based filtering does not compute similarity between two users. Content-based filtering has a huge advantage in a system with very few users. In a content-based RS, a newly added item can be recommended when no users have rated. The recommendation depends on the similarity between items instead of users.

For new registered user, the content-based RS cannot recommend anything. It requires at least one rating from the user to know what the preferences of the user.

The detailed data about an item may interfere the RS to make correct recommendations. For example, actors and directors in a movie are not an interesting factor in some cases [2].

2.2.3 Demographic-based filtering

Demographic-based filtering recommends items by knowing user profiles [3]. The data often include aspects such as age, gender, occupation and living region. These aspects will affect users' preferences, tastes, purchasing habits, purchasing power, etc.

2.2.4 Challenges in Recommender System

Cold start and long tail are two common problems in RS. Cold start refers to a RS cannot function correctly when there are too few users. Long tail is a phenomenon that a small proportion of items are extremely popular and the rest of them are unpopular.

2.2.4.1 Cold start

Cold start is a common problem in RS. A RS with too few users is almost impossible to make recommendations. If the user/items ratio is small, even if users rated a few movies, it is highly unlikely to find items they involved in common. Thus the system is not able to recommend any items [4]. The simplest method to address this problem is to recommend the most popular items and "warm up" the RS. When some ratings are collected, one of the approach mentioned above can be applied and RS is able to operate normally.

2.2.4.2 *Long tail*

The phrase *long tail* was firstly mentioned by Clay Shirky [5]. Long tail is a phenomenon which refers to extremely high involvement of a small amount of items. On the other hand, the involvements of the rest of the items are very low. These items with low involvement occupy a large proportion. For e-commerce companies, such as Amazon, most of the items have a low number of sales. For a movie streaming service, most of the movies are viewed very few times. Although unpopular items are consumed very few times, but because of its high proportion, it is a curial for the revenue.

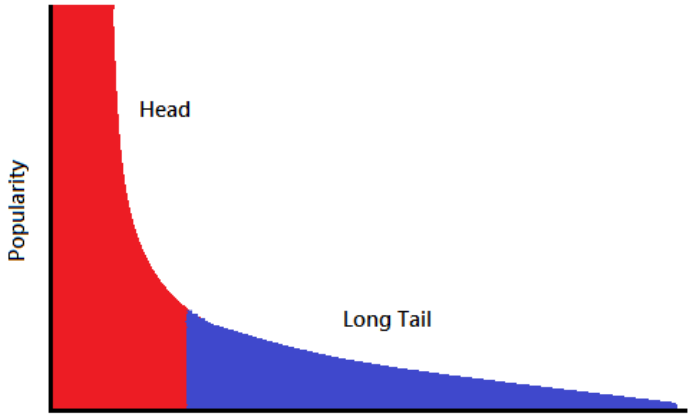

Figure 2.1: Distribution of item consumption amount of the long tail phenomenon.

2.2.4.3 *Long tail in collaborative filtering*

As mentioned above, collaborative filtering finds items what users have commonly involved. The number of ratings received for popular items are much higher than those of unpopular items. This will inevitably lead to a problem that the system is more likely to recommend popular items. Accordingly, it will lead to popular items even more popular and unpopular items even less popular. This is a phenomenon called Matthew Effect [6]. One of the solutions to this problem is to appropriately reduce the popularity of the most popular items, in order to give more opportunities for unpopular items to appear on the recommendation list.

2.3 Data collection

To make personalized recommendations, algorithms in RS nearly always require large amount of data. Therefore, data collection is the most important step in building a RS.

There are different kinds of methods to choose when collecting data. The choice of methods often depends on what need to be recommended. The most common solution is to create a web crawler which finds data from the Internet. However, this solution is not applicable for every case. Some companies need data from their customers. Another solution is to use company collected data which contains infor-

mation about its customers. The last solution is to use a dataset collected by a third party.

2.3.1 Feature

In RS, a feature corresponds to a column of a table in database. In RS related mathematics, each feature is often represented by a dimension in a matrix. An object includes two or more features can be represented in corresponding dimensions in mathematics.

2.3.2 Ratings

In order to give the most accurate predictions, it is needed to know what users think about the recommended items. This can be achieved by a survey or items often browsed by users. For recommender system to analyze, these results have to be converted into numbers. Hence, the most accurate and efficient way to know what users think about a product is to let users directly set score on items.

2.3.2.1 Explicit ratings

Explicit ratings are scores which users rate. It shows what users think in a subjective manner. A common problem with explicit ratings is users do not rate. A user may purchase a large number of items without rating any of them. A recommender system which only takes explicit ratings as input is not able to predict what users may like in this situation.

2.3.2.2 Implicit ratings

Implicit ratings are calculated by observing the behavior of users. It can be known such as the total time a user spends on a web page; how many percent of a song was listened before switched to next etc. [7]. These behaviors must be measurable and represented by numbers. These numbers which describe users' behaviors are features. A specially designed algorithm calculates these numbers and gives an output in a single number as rating. Usually, each feature is not equally important as related to results, thus the algorithm gives a weight for each feature depending on its importance. The weight is equivalent to coefficient in mathematics.

2.3.2.3 Rating scale

Rating could be a scale that covers a range of 1 - 10, 1 - 5, or -1 - 1. The 1 - 5 rating scale may represent from lowest to highest: hate, dislike, neutral, liked and love. The 1 to 10 rating scale is relatively detailed in comparison to 1 - 5.
From algorithmic and statistical perspective, choosing 1 - 10 instead of 1 - 5 increases the differentiation between users' opinions. It will make data processing more convenient. However, users rate in 1 - 10 scale are more likely hesitant which leads to set a biased score or even not to rate at all. A binary rating system uses like/dislike (also known as up/down vote). This is represented mathematically -1 - 1. It is the simplest choice available in explicit ratings. Users would not put too much thought about a score.

In movie rating or video rating, each company chose a different rating scale than others. IMDb uses 1 to 10, Netflix chose a 5-star rating system, which is essentially 1 - 5 [8], and YouTube changed their 5-star rating system to a like/dislike system in 2009. The reason of YouTube changed its rating scale to like/dislike because it was

discovered that users on YouTube tend make choice of “all or nothing” instead of a scale of 1 - 5 [9].

2.4 Data preprocessing

Data preprocessing is one of the most important tasks for data analysis problems. In order to ensure simplicity and efficiency of algorithms, the data often needs to be extracted before it feeds into algorithms. To save memory, irrelevant columns in datasets will be removed. This gives a significant performance improvement for large datasets [10].

2.4.1 Similarity metrics

Similarity is a float number which describes affinity of two objects. This section explains three similarity measuring algorithms and discusses its advantage and disadvantage. In all of the formulas presented below, n represents the number of elements of each vector. Each of A and B represent a vector with same amounts of elements.

2.4.1.1 Manhattan distance

Manhattan distance is also known as *taxicab geometry*. It measures distance between two points by adding the absolute differences.

$$\text{manhattan}(\mathbf{A}, \mathbf{B}) = \sum_{i=1}^{n} |A_i - B_i| \tag{1}$$

The formula can be applied to a system with dimensions ≥ 2.

The advantage and disadvantage of manhattan distance:
For example, when measuring similarity between users, a dataset contains ratings from users. If an item is not rated by a specific user, the rating is set to zero by default, often called missing value. The distance cannot be accurately measured using Manhattan distance if there are too many missing values [11].

2.4.1.2 Cosine similarity

Cosine similarity calculates cosine of an angle between two objects. The range is in [1, -1]. If two vectors describe the same direction, the cosine similarity is equal to 1. If two vectors at an angle of 90°, the cosine similarity is equal to 0. And if two vectors describe the opposite direction, the cosine similarity is equal to -1 [12].

The dot product of A and B:

$$A \cdot B = \sum_{i=1}^{n} A_i \times B_i \tag{2}$$

The magnitudes of A and B:

$$||A||\,||b|| = \sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2} \tag{3}$$

Combining them together gives:

$$\text{cosine}(\mathbf{A}, \mathbf{B}) = \frac{A \times B}{||A||\,||b||} \tag{4}$$

In real-life situations, users always rate a small subset of all items. Thus it is most likely to see two users did not rate anything in common. In this case, cosine similarity has the advantage to calculate similarity accurately. Furthermore, cosine similarity ignores zeros (not rated items) in calculation.

2.4.2 Dimensionality reduction

As mentioned in 2.4, to receive personalized recommendations for each user, the matrix (dataset) must be large. If a matrix has 2 dimensions or higher and the length of each dimension is huge, RS need to process a large amount of data. The would lead to high RAM consumption and a long processing time.

Dimensionality reduction is a range of approaches to reduce dimensions of matrix by decomposing a big matrix into smaller matrices [13]. These approaches would save huge amount of memory and increase the performance of computing recommendations. In addition, dimensionality reduction is also able to remove redundant data.

2.4.2.1 SVD

SVD, or singular value decomposition is one of the approach to reduce dimensions [14]. The core of the formula is as follows:

$$\mathrm{M} = \mathrm{U}\sum\mathrm{V}^T \tag{5}$$

Where M is the matrix to be decomposed, U, ∑, V are decomposed matrices. ∑ is a diagonal matrix; U and V are both unitary matrixes. In mathematics, a unitary matrix is defined as:

$$U^T U = UU^T = I \tag{6}$$

Where I is an identity matrix. An identity matrix consists of ones on its main diagonal and the rest of positions are filled with zeros.

An example of a 3x3 identity matrix:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

After decomposition, the most important features are stored in the eigenvector ∑. The number of rows and columns of an identity matrix is always same, usually represented by *k* or use the name of "k-value".

Decomposing the matrix M neither solves the goal of saving RAM nor reduces processing time. However, since the matrix is decomposed and the eigenvector ∑ is containing the most important features, by using a small part of ∑ could restore the most part of the matrix M. It means that the most of ∑ can be discarded. P. Klein [15] pointed out that by retaining 5% of the k-value, more than 95% of values would be correctly restored. According to the rules of matrix multiplication, when the size of ∑ is reduced, U and V^T are also going to be reduced correspondingly. For example, in a practical situation:

- a = 1000, which is number of users
- b = 6000, which is number of movies. When reduced to 5%, the new value is equal to 300.

The size of the original matrix would be:

Table 2.1 A comparison of matrices U, ∑ and V^T before and after reduction.

Matrices	**Size before reduction**	**Size after reduction**
U	1000 x 6000	1000 x 300
∑	6000 x 6000	300 x 300
V^T	6000 x 6000	300 x 300

After the reduction, matrices have been largely reduced and the goal of saving RAM and processing time are achieved.

2.5 Methods of splitting dataset into training set and test set

After a RS is implemented, algorithms are needed to process as much data as possible to make recommendations accurate. The quantity of data and the accuracy of prediction are in direct proportion [16]. The process of feeding data into algorithm to improve its accuracy is called training.

2.5.1 N-fold cross validation

N-fold cross validation is used to evaluate classification or regression algorithms. In N-fold cross validation, a dataset is randomly divided into N equally large subsets. Then N-1 subsets are going to be used to train the algorithm and the last subset is reserved to evaluate its accuracy. The most commonly chosen number of N is 10, which divide a training set into 10 equally large subsets, 9 of which for training and reserve the last one for evaluation.

The N-fold cross validation is a non-deterministic evaluation method. Since subsets are randomly divided, the possibility to repeatedly obtain the same result is highly unlikely. These results, however, conform to normal distributions in statistics.

2.5.2 Split dataset

A dataset may also be split with a certain proportion. If a training set is too large after divided, it will leave a small quantity of data for evaluation. On the contrary, if the training set is too small, the training result would be poor which will inevitably lead to a poor evaluation result.

In the aspects of evaluation result, split dataset has the same characteristic as N-fold cross validation. Since dataset is randomly split, the result is non-deterministic but conform to normal distribution in statistics.

2.5.3 Leave-p-out

Leave-p-out is a method which reserves p rows of data as test set (evaluation data) and train algorithms with the rest of data. This process repeats until every row is used as evaluation data. This means that for a large dataset, leave-p-out is much more time consuming than splitting the dataset with a certain ratio. The complexity for leave-p-out is $O(n^{n/p})$, and complexity for split dataset is O(n) [17].

2.6 Evaluation

After each training phase, predictions and recommendations are made by algorithms. The recommendation need to be evaluated in order to know its accuracy. Two commonly used algorithms, precision and recall are presented below. Moreover, I have created an improved version of the precision algorithm to address the evaluation bias in a situation when the predicted ratings exceed the maximum allowable rating.

Since the predicted ratings nearly always presented in decimals and the real ratings are always in integer. It is highly unlikely to have exactly same value of predicted ratings and real ratings. Additionally, if the predicted rating is not far from real ratings, it should be considered as an accurate prediction. For these reason, cutoff value is always applied as a parameter in these algorithms to create a tolerance of prediction bias in a certain range.

2.6.1 Precision

Precision and recall are two algorithms which evaluate the recommendation accuracy.

The following four cases are indicator of evaluations:

- True Positive (TP): RS recommends an item which is favored by the user.
- False Positive (FP): RS recommends an item which is not favored by the user.
- True Negative (TN): RS does not recommend an item which is not favored by the user.
- False Negative (FN): RS does not recommend an item which is favored by the user.

The precision formula is defined as [18]:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{7}$$

Precision measures the fraction of recommended items over all items which are recommended.

2.6.2 Recall

Recall measures the fraction of recommended items over all items which is favored by the user [18]:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{8}$$

2.6.3 Improved Precision algorithm

Since the predicted ratings may exceed the maximum rating, an improved version of the precision algorithm was created to prevent this situation. In the improved precision algorithm, if a predicted rating exceeds the allowable maximum rating, the predicted rating would be assigned to the maximum rating. For example, a predicted rating for a movie is 5.23 and the maximum rating is to 5. Because of the predicted rating is higher than 5, it would be assigned to 5. The rest part of the algorithm is the same as the original precision algorithm. For all the RS which would produce predicted ratings higher than the maximum rating, the Improved Precision algorithm gives a better reflection of precision than the original one.

3 Methods

This chapter describes the selection of recommender system, how data collection was proceeded, method of preprocess data and selected evaluation metrics.

3.1 Overview of open source recommender systems

There are various kinds of RS provided by the open source community [19]. These RS are either built on cluster computer frameworks such as Apache Spark [20] or built without cluster computer framework. RS built on cluster computer frameworks are able to handle massive data (commonly called Big Data). RS built without a big data framework only handle a relatively small amount of data. In addition, these RS are implemented by various kinds of programming languages, such as Java, Scala [21], Python [22], Ruby [23] and JavaScript [24] with Node.js [25].

Table 3.1: A list of RS using different big data frameworks and programming languages.

Name of project	Big data framework or Database management system	Implementing programming language
PredictionIO	Apache Spark	Scala
Racoon Recommendation Engine	None	JavaScript (Node.js)
HapiGER	None	CoffeeScript (Node.js)
Mahout	Hadoop [26]	Java/Scala
Seldon	Apache Spark	Java
LensKit	None	Java
Oryx v2	Apache Spark	Java
RecDB	PostgreSQL	C
Crab	None	Python
Predictor	None	Ruby
Python-recsys	None	Python

3.2 Selected open source recommender system

There are many aspects to consider when selecting an open source RS. PlayPilot requires that the selected RS has to meet the following criteria:

- code readability
- maintainability
- implementing programming language
- the features of data collected by PlayPilot is matching the implementation approach of RS
- documentation readability

3.2.1 Code readability, maintainability and implementing programming language

In order to ensure code readability and maintainability, PlayPilot requires the recommender system to use the same programming language as their web application. Since the web application of PlayPilot is implemented in python, PlayPilot requires its recommender system also written in the same language. Since the selected RS must be implemented in python, there are only two open source RS available to choose. These two RS are called *python-recsys* [27] and *crab* [28].

3.2.2 Implementation approach and data collection

The sort of features of data owned by PlayPilot was going to play a decisive role when selecting RS (implementation approach). To select the ideal approach to a RS, two aspects related to data need to be considered, features and quantity. As described in section 2.3.1, features are data which describe an object from different aspects. By having diverse of features collected, it gives a variety of options to select different implementation approaches. This does not mean that all of the collected features must be used.

Data quantity refers to the amount of data. The requirements of data quantity in statistics and RS are not exactly the same. In statistics, when the quantity of sampling reaches a certain threshold, the distribution is basically remains the same. Increasing the quantity of sampling will create a more accurate distribution, but the trend remains the same. However, for a RS which has to provide personalized recommendations, the dataset must be as large as possible. With a small data quantity, the accuracy of predictions would be exceedingly poor, which leads the recommendations totally meaningless. The features required for different filtering approaches are listed below:

Table 3.2: Different filtering approaches and its corresponding features required.

Approaches	Type of data required	Examples in a movie recommender
Content-based filtering	Information about products	Title, belonging genres, weights of genres, country of origin, year of release
Collaborative filtering	Users feedback about products	Rating, watched, shared
Demographic-based filtering	Information about users	Age, gender, occupation, living region

Since the filtering approach is implemented by the RS, the selection of filtering approach is essentially based on which RS is selected. Both *python-recsys* and *crab* uses SVD, which is a branch of collaborative filtering. It means that collaborative-based filtering was selected as the filtering approach.

3.2.3 Open source licenses

Each open source software uses an open source software license to promote their software as well as protect their copyrights. Some open source software licenses are allowing commercial user while others not. To choose an open source RS for PlayPilot which they can use in production, the license of the RS must allow commercial use. There are only two RS left to select and these two use different licenses. *Crab* is using BSD as their license and *python-recsys* applied GPL.

3.2.3.1 Berkeley Software Distribution licenses

Berkeley Software Distribution licenses or BSD licenses allows commercial use and modification [29].

3.2.3.2 General Public License

General Public License or GPL [30] has similar terms as BSD. The difference is that the modified version of a GPL software must still use GPL. This means that the modified code must be disclosed to the public. Additionally, the significant change made to GPL software must be stated.

Fortunately, both licenses allow commercial use. This means that both Crab and python-recsys have not been eliminated due to licenses are not appropriate.

After some research, python-recsys is selected based on the fact that its documentation is concise and efficient to read and has a clear code structure. *Crab* is relatively worse in these areas.

3.2.4 Selected features

The data collection for PlayPilot consists of the following for each movie, TV series, documentary and other TV program:

- Title
- Genres, may have more than one
- Year of release
- IMDb ID
- IMDb rating
- Country of origin

Furthermore, PlayPilot has collected the following data for each user:

- Year of birth
- Gender
- Add a movie to favorite
- Whether has seen a title or not

However, above data do not meet the need of a RS. Thus PlayPilot had decided to collect other user data during this project. These data include:

- Ratings in a scale of 1 to 10
- Save titles to a watch list
- Mark a title as not interested

Because all RS are using explicit ratings as an indicator, it is axiomatic to use explicit ratings (in this case, scale 1 to 10) as input data.

3.3 Selected dataset

PlayPilot was collecting movie ratings from its users during this project. However, the growth of data collected by PlayPilot cannot meet the demand of this project. Therefore, a dataset collected by a third party is required. A dataset collected by MovieLens [31] ("the MovieLens dataset") was selected. The MovieLens dataset is collected by GroupLens Research of Minnesota University through their lab affiliated nonprofit website. The research team made it publicly available on the Internet. There are five datasets, each containing different quantity of data. Details of these datasets are listed as below:

Table 3.3: Datasets provided by MovieLens containing different quantity of users, movies and user ratings on movies

Name of datasets	No. of users	No. of movies	No. of ratings
MovieLens 100K	1,000	1,700	100,000
MovieLens 1M	6,040	4,000	1,000,000
MovieLens 10M	72,000	10,000	10,000,000
MovieLens 20M	138,000	27,000	20,000,000
Latest	240,000	33,000	22,000,000

After a series of tests, the dataset MovieLens 1M is selected. This decision was made due to the limited timeline of this thesis work. There are numbers of different k-values and cutoff values to be tested (evaluated). To simulate with all these k-values and cutoff values with a dataset larger than MovieLens 1M, the processing time would grow exponentially.

3.4 Data preprocessing and recommendation

By reconstruct the rating matrix using U, ∑, and V^T, all the zeros in the original matrix will be filled. The values which filled in are the predicted ratings. The formula is defined as:

$$M' = UkV^T \tag{9}$$

The k is the k-value, which represents the first k dimensions of eigenvalues in the ∑ matrix.

The size of k-value will affect the recovery rate of the matrix. To be able to focus on the evaluation of recommendation accuracy, the recovery rate for different k-values will not be measured. (However, different k-values will be tested for the result of recommendation accuracy, which is the most important part in this thesis). Since the selection of k-value would affect the recovery rate of the matrix, which would further affect the recommendation accuracy, the selection of k-value is still very important. As mentioned in section 2.4, if k-value is 5% of the number of rows/columns of matrix ∑, more than 95% of values would be correctly restored. Since the selected dataset has 6040 columns, 5% of 6040 is approximately equal to 300.

The most time consuming part of the simulation is decomposing the matrix and k-values determines the decomposing efficiency. In addition to 300, two smaller numbers, 100 and 200, were selected to evaluate the RS. The selection is based on the expectation to find the best tradeoff between decomposing efficiency and recommendation accuracy. The results produced by three k-values, 100, 200 and 300 will be shown and discussed in sections 4 and 5.

3.5 Selected splitting method

As mentioned in section 2.5, to divide a dataset into a training set and a test set, there are three methods. The selection method is curial for training result and ultimately affects evaluation results. Improper split ratio would influence the result negatively. If a training set is too large after divided, it will leave a small quantity of data for evaluation. On the contrary, if the training set is too small, the training result would be poor which will inevitably lead to a poor evaluation result.

In order to find out the accuracies of recommendation lists, these evaluations will only compare the same movie rated by the same user in the top n lists (in this case, n is equal to 10) of all users and the test data set. Since the probability to match is very low, the ratio of test set must be larger than other evaluations. Therefore, a few python scripts are written to find out which ratio is appropriate. To be appropriate, it means that when the split ratio of test set reaches a certain threshold, the probability to find at least one matched movie rated by the same user is 100%. After a

series of test, it shows that 80/20 is an appropriate split ratio. In other words, a dataset needs to be split into 80% for training and 20% for test in order to guarantee there is something evaluable.

3.6 Selected evaluation algorithms

The evaluation of the selected RS focus on whether the users are satisfied with recommendations or not. *Precision* and *improved precision* was selected to evaluate the selected RS. These two algorithms compute the recommendation accuracy which may be further interpreted as users' satisfaction rate.

3.6.1 Selected cutoff values

Cutoff values indicate how much deviation of ratings users normally allow. Small cutoff values lead the recommendation accuracies seem lower. High cutoff values indicate that users allowing larger differences between predicted ratings and their real ratings. Therefore, only a range of moderate cutoff values produce meaningful evaluation of recommendation accuracy. The maximum allowable rating is 5. It should be reasonable to set the maximum cutoff value 10% of 5, which is 0.5. The cutoff values lower than 0.5 are also worth evaluating, since by comparing evaluation results using different cutoff values as parameters show a clear picture of the relation between cutoff values and evaluation results. Therefore, five cutoff values, 0.1, 0.2, 0.3, 0.4 and 0.5 were selected to evaluate the RS.

3.7 APIs of python-recsys

The APIs of python-recsys provide the most important functionality, decompose the rating matrix and recommend. This section covers these core APIs as well as gives a simple introduction of other APIs, such as evaluations.

3.7.1 Used APIs for this thesis work

Name: split_train_test(percent)
Split a dataset into training set and test set.

Parameter:

percent - The percentage of training set. The rest (100% - **percent**) are reserved for test set.

Name: compute(k, min_values)
Decompose the rating matrix into three matrices.

Parameters:

k – the k-value, which represents how many dimensions of $\sum$ will be reserved. The rest will be discarded.

min_values - remove rows or columns from the rating matrix that has less than "min_values"

Name: recommend(userid, n)
Recommend n movies for the given user.

Parameters:
userid – The ID of the user.

n – The number of recommended movies. The highest predicted ratings show first.

3.7.2 Other available APIs

The APIs for evaluations was not used due to it only provide the evaluation for matched items. The code to match items in recommendation lists and items in the test set was not implemented. To accomplish the evaluations in this thesis work, the evaluation API do not meet the demand, therefore cannot be used [32].

4 Results

Two algorithms, precision and improved precision are used to evaluate the recommendation accuracy. Each algorithm is evaluated with five cutoff values, 0.1, 0.2, 0.3, 0.4 and 0.5. Each evaluation is going to be repeated with three k-values. Along with these five cutoff values, these two algorithms will be measured with k-values (dimensions of eigenvalues) 100, 200 and 300. As mentioned in 3.5, 80% from the dataset are used for training and the rest of 20% reserved to evaluate the recommendation accuracy. Since the training set and test set was split randomly, the recommendation result is characterized as non-deterministic. Therefore, 50 simulations were executed along with these five cutoff values and these three k-values. The mean values of these simulations are shown below:

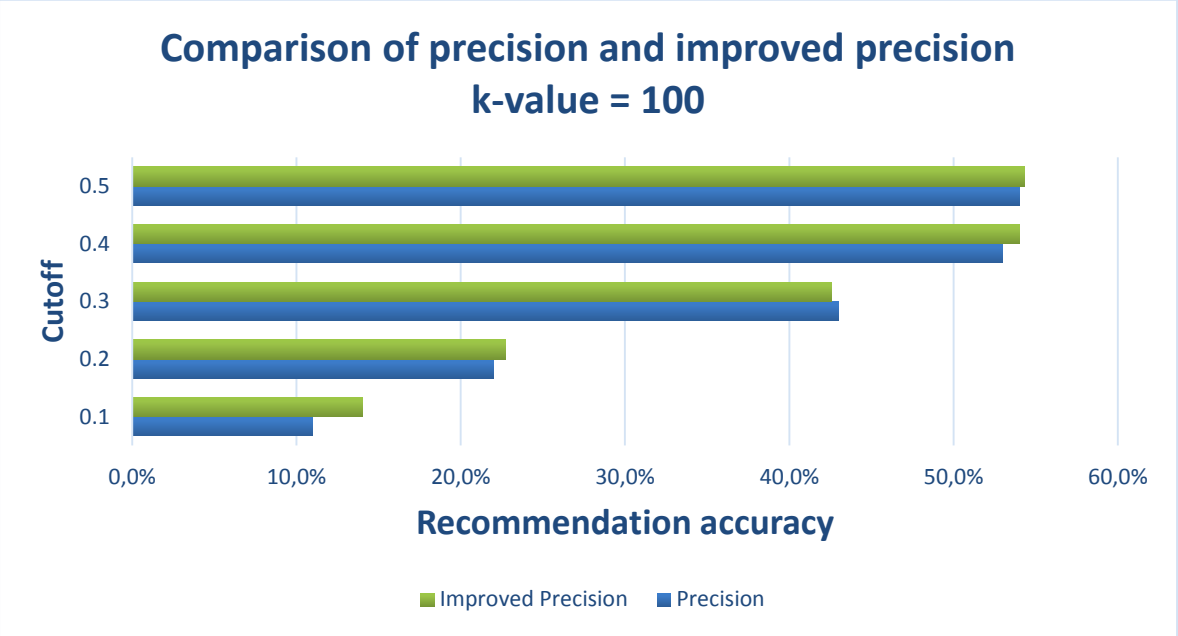

Figure 4.1: A comparison of recommendation accuracy of two algorithms, precision and the improved precision. The recommendation accuracies are computed with cutoff values 0.1 – 0.5, using k-value of 100.

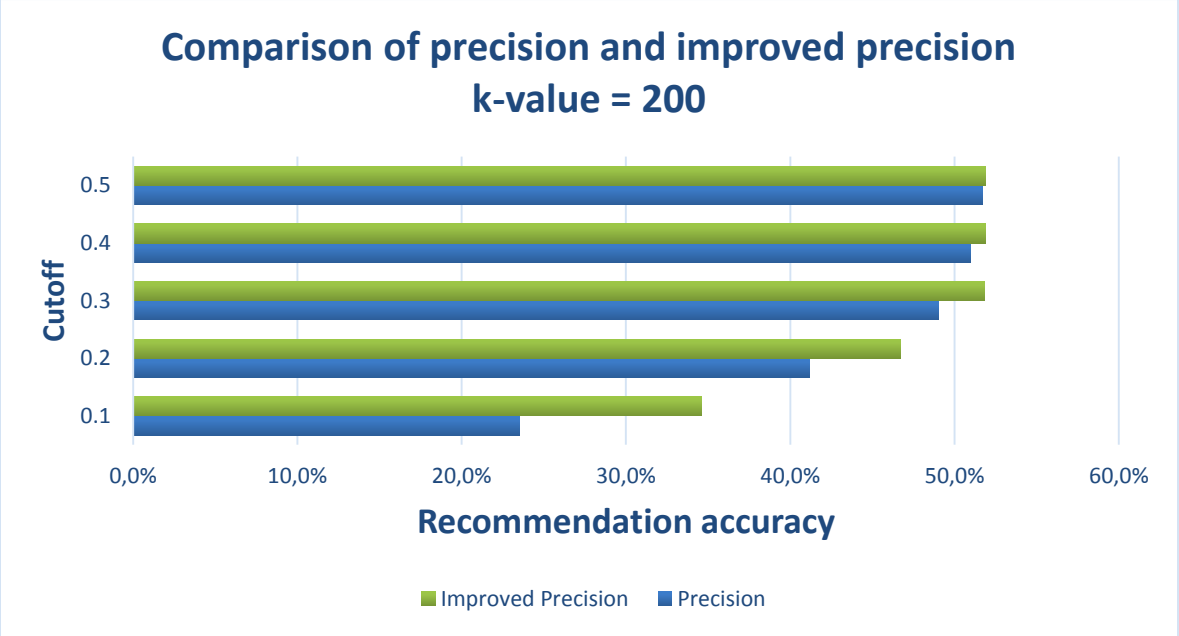

Figure 4.2: A comparison of recommendation accuracy of two algorithms, precision and the improved precision. The recommendation accuracies are computed with cutoff values 0.1 – 0.5, using k-value of 200.

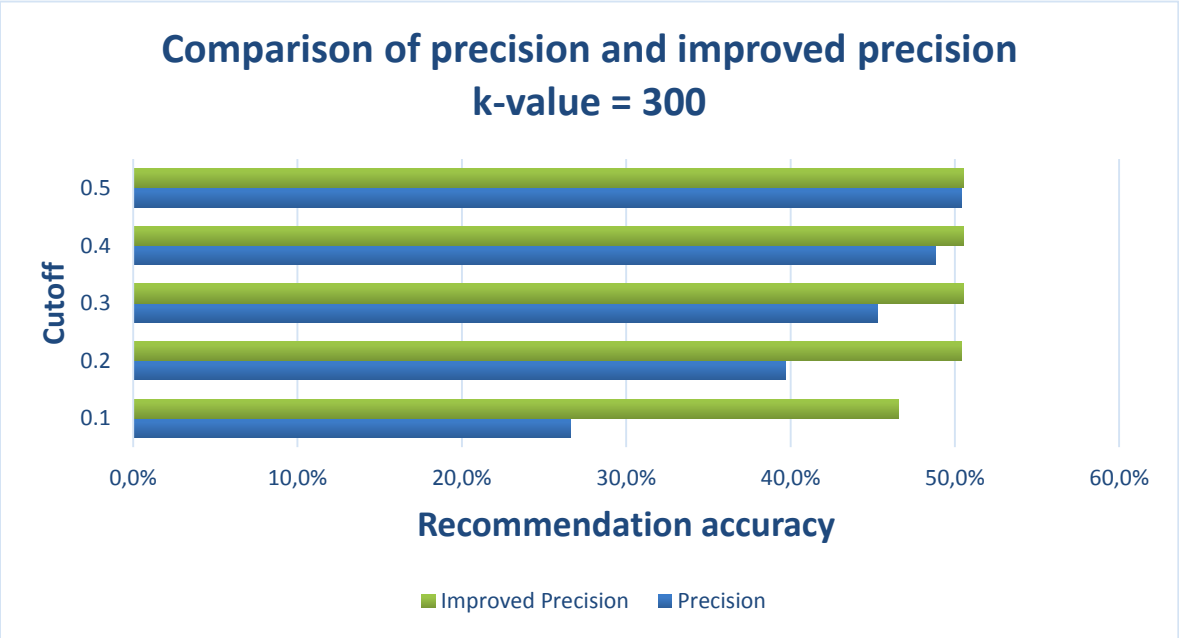

Figure 4.3: A comparison of recommendation accuracy of two algorithms, precision and the improved precision. The recommendation accuracies are computed with cutoff values 0.1 – 0.5, using k-value of 300.

5 Analysis and discussion

This chapter presents the hypothesis. In addition, this chapter also analyzes and discusses the results and how the thesis work will improve the world in sociology and economics perspectives.

5.1 Hypothesis and interpretation of results

It was hypothesized that the evaluation results of the improved precision algorithm are higher than the original one. This hypothesis was confirmed by the evaluations. It was also hypothesized that the recommendation accuracies are higher (when comparing same algorithm and cutoff value) when k-value increases, but the results show that this is only true for cutoff values 0.3 or lower. It proves that the k-value does affect the recommendation accuracies of the original matrix, but cutoff values have a greater impact on recommendation accuracies than of k-values, which is not surprising. The unexpected part is when cutoff values reaches 0.4 and 0.5, the recommendation accuracies are even higher in lower k-values. The reason of this phenomenon is still unclear.

Since the top n list is sorted by predicted ratings in descending order, most of the predicted ratings are more likely to have higher ratings than the real ratings. By running 50 simulations with these two algorithms, the results showed that the improved precision algorithm is slightly higher than the original precision algorithm.

5.2 Limitations of SVD

Because SVD is a collaborative filtering algorithm, it requires many users to participate. When the user/item ratio is small, SVD is going to make poor recommendations. However, collaborative filtering is the least bad choice. Recommendations produced by other approaches may have even greater bias.

The collaborative filtering is prone to recommend popular movies and make them even more popular. This will lead the unpopular movies have even less chance to appear on the recommendation list. As mentioned in 2.2.4.3, it is possible to address this problem by lower the ratings of popular movies. But this solution will sacrifice the recommendation accuracy which lowers the satisfaction rate from users. Since the goal of RS is to provide the best possible service for users, this solution is unnecessary.

5.3 Sustainable development of society, economy and environment

By using SVD, which is a collaborative filtering algorithm, the quantity of users and quantity of ratings are directly proportional to the accuracy of predictions. It means that the satisfaction rates will be increased when more movie audiences registered on the website and they rate more. This is consistent with the Matthew Effect mentioned in section 2.2. The author believes that this RS will allow PlayPilot to attract new users and retain existing users. Furthermore, to some extent, the RS will also increase profit and revenue for PlayPilot.

This project only requires a computing system with low energy consumption. The most processing consumption is where the decomposition of rating matrix (SVD) take place. The matrix decomposed in this project has a size of 1,000,000 rows and 6040 columns with a density of 4.47%. Depending on k-values 100 to 300, it takes 60 to 70 seconds to decompose the matrix on an Intel Core i5 4210u processor. Whenever the new ratings are provided and the matrix is required to be decomposed again, it consumes only a small amount of electricity. Based on the fact that it only takes 60 to 70 seconds to recompute on a normal PC processor, the energy consumption is extremely low. The frequency of recomputation could be an important factor of energy consumption, but it is still undecided yet. Since the energy consumption is very low, even more frequent recalculations will not have any big environmental effect.

The low energy consumption is owing to the mathematical techniques provided by SVD. Noises (redundant data) have been removed in order to optimize the computing efficiency. Taking into account of the positive influence on society and economy provided by this RS, such low energy consumption is very worthwhile.

6 Conclusion

This thesis work, an improved version of precision algorithm, was created to evaluate the open source RS, *python-recsys*. The results produced by the improved precision algorithm definitely present a better reflection of the recommendation accuracies.

6.1 Recommendation

The goal of this thesis work is achieved, and it is recommended for PlayPilot to use the improved version of precision algorithm to evaluate the results of the selected RS. Since there is a possibility that the predicted ratings exceed the maximum allowable rating, the improved precision algorithm is also applicable to other RS which have the same characteristic.

6.2 Improvement possibilities and future work

Both collaborative filtering and content-based filtering have their advantages and limitations. A RS using SVD, a collaborative filtering algorithm, was selected by this thesis work. It is possible to implement other collaborative filtering algorithms such as slope one or adjusted cosine similarity. Additionally, it is possible to implement hybrid filtering approaches by combining collaborative filtering with content-based filtering, or combining collaborative filtering with demographic-based filtering. It is important to note that combining more than one filtering approach requires more features of data.

Some of the data is difficult to collect. For example, it would produce significant recommendations by collecting data about each user's rating on each genre. Based on these ratings, predicted weights of genres of each movie could be computed using algorithm A. Then, algorithm B, which is an inverse operation of algorithm A, reproduces more accurate user ratings on each genre. And then uses algorithm A to reproduce weights of genres of each movie and so on. The process can repeat many times and the results will gradually converge. Using this approach, very accurate and realistic results can be produced when this process repeats many times [33]. This approach requires large RAM and long computation time which is beyond the processing capability for a PC. Hence a computer cluster is needed to solve this problem.

References

[1] Aalto E. Learning Playlist Representations for Automatic PlayList Generation. Stockholm:, Computer Science and Communication; 2015.

[2] Ricci F, Rokach L, Shapira B, Kantor PB. Content-based Recommender Systems:State of the Art and Trends. In Lops P, Gemmis Md, Semeraro G. Recommender Systems Handbook.: Springer; 2010. p. 73-100.

[3] Vozalis M, Margaritis K. Collaborative filtering enhanced by Demographic correlation. Thessaloniki: University of Macedonia, Applied Informatics.

[4] Gorakala S, Usuelli M. Limitations of collaborative filtering. In Building a Recommendation System with R.: Packt; 2015. p. 73.

[5] Dix A, Levialdi S, Malizia A. Semantic Halo for Collaboration Tagging Systems. ; 2006.

[6] Perc M. The Matthew effect in empirical data. Maribor: University of Maribor, Faculty of Natural Sciences and Mathematics; 2014.

[7] Carlquist O, Boström Leijon S. Implikat - A System for Categorizing Products using Implicit Feedback on a website. Degree project. Stockholm:, School of Technology and Health; 2014. Report No.: 2014:24.

[8] Netflix Ratings & Recommendations. [Online]. [cited 2016 03 30. Available from: https://help.netflix.com/en/node/9898.

[9] YouTube Official Blog. [Online]. [cited 2016 03 30. Available from: https://youtube.googleblog.com/2009/09/five-stars-dominate-ratings.html.

[10] Garcia S, Luengo J, Herrera F. Data Preprocessing in Data Mining: Springer.

[11] Zacharski R. Collaborative filtering. In A Programmer's Guide to Data Mining.; 2014. p. 20-124.

[12] Sidorov G, Gelbukh A, Gomez-Adorno H, Pint D. Soft Similarity and Soft Cosine Measure: Similarity of Features in Vector Space Model. ; 2014.

[13] Richert W, Coelho LP. Dimensionality Reduction. In Building machine learning system with python. Birmingham: Packt Publishing; 2013. p. 221-225.

[14] Sarwar B, Karypis G, Konstan J. Incremental Singular Value Decomposition Algorithms For Scalable Recommender Systems. Minneapolis: University of Minnesota, Computer Science and

Engineering.

[15] Klein P. Coding the Matrix: Linear Algebra through Applications to Computer Science: Newtonian Press; 2013.

[16] Banko M, Brill E. Scaling to Very Very Large Corpora for Natural Language Disambiguation. Redmond:; 2001.

[17] Celisse A. Optimal Cross-validation in Density Estimation With the L-LOSS. The Annals of Statistics. 2014; 42: p. 34.

[18] Vergne M. Mitigation Procedures to Rank Experts through Information Retrieval Measures. Trento:, Center for Information and Communication Technology; 2016.

[19] Jenson G. A list of Recommender Systems and Resources. [Online]. [cited 2016 05 03. Available from: https://github.com/grahamjenson/list_of_recommender_systems#open-source-recommender-systems.

[20] Spark. Apache Spark. [Online]. [cited 2016 05 03. Available from: http://spark.apache.org/.

[21] Scala-lang.org. The Scala Programming Language. [Online]. [cited 2016 05 20. Available from: http://www.scala-lang.org/.

[22] Python.org. About Python | Python.org. [Online]. [cited 2016 05 20. Available from: https://www.python.org/about/.

[23] Ruby-lang.org. Ruby Programming Language. [Online]. [cited 2016 05 20. Available from: https://www.ruby-lang.org/en/.

[24] Mozilla. JavaScript | MDN. [Online]. [cited 2016 05 20. Available from: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

[25] Node.js Foundation. Node.js. [Online]. [cited 2016 05 20. Available from: https://nodejs.org/en/.

[26] Foundation AS. Welcome to Apache Hadoop. [Online]. [cited 2016 05 20. Available from: http://hadoop.apache.org/.

[27] Celma O. Python-recsys library. [Online]. [cited 2016 05 16. Available from: http://ocelma.net/software/python-recsys/build/html/index.html.

[28] Caraciolo M, Melo B, Caspirro R, Vieira R. Recommender System Framework in Python. [Online]. [cited 2016 05 16. Available from: http://muricoca.github.io/crab/.

[29] The BSD 2-Clause License. The BSD 2-Clause License | Open Source Initiative. [Online]. [cited 2016 05 19. Available from: https://opensource.org/licenses/bsd-license.html.

[30] GNU General Public License. The GNU General Public License v3.0 - GNU Project - Free Software Foundation. [Online]. [cited 2016 05 19. Available from: http://www.gnu.org/licenses/gpl-3.0.en.html.

[31] MovieLens. MovieLens. [Online]. [cited 2016 05 03. Available from: http://grouplens.org/datasets/movielens/.

[32] Celma O. API - python-recsys v1.0 documentation. [Online]. [cited 2016 05 30. Available from: http://ocelma.net/software/python-recsys/build/html/api.html.

[33] Holehouse A. Recommender systems - introduction. [Online]. [cited 2016 05 22. Available from: http://www.holehouse.org/mlclass/16_Recommender_Systems.html.

TRITA 2016:51