An Open Science Approach to Exploring Time-Accuracy Trade-offs in Recommender Systems

by

Khalid Jahangeer

(Under the Direction of John A. Miller)

Abstract

Recommender Systems have become an integral part of our consumer dominated world. With the evolution of Big Data and the exponential expansion of consumerism it is imperative that we design efficient recommender systems. Collaborative Filtering techniques have been popularly adopted by researchers for developing robust recommender systems. In this paper we discuss some of the techniques that fall under the Collaborative Filtering umbrella and have been implemented within the ScalaTion big data analytics framework. Apart from discussing the implementation we analyze the execution time and accuracy of these techniques. This analysis has been performed to explore trade-offs between time and accuracy that occur while performing predictions using these techniques.

Index words: Recommender Systems; Collaborative Filtering; Matrix Factorization; Singular Value Decomposition;

An Open Science Approach to Exploring Time-Accuracy Trade-offs in Recommender Systems

by

Khalid Jahangeer

B.S., Amity University, 2011

A Thesis Submitted to the Graduate Faculty
of The University of Georgia in Partial Fulfillment
of the
Requirements for the Degree

Master of Science

Athens, Georgia

2017

An Open Science Approach to Exploring Time-Accuracy Trade-offs in Recommender Systems

by

Khalid Jahangeer

Major Professor: John A. Miller

Committee: Krzysztof J. Kochut
Hamid R. Arabnia

Electronic Version Approved:

Suzanne Barbour
Dean of the Graduate School
The University of Georgia
December 2017

Acknowledgments

Firstly, I would like to extend my heart-felt gratitude towards my major professor Dr. John A. Miller. His constant guidance and never ending support has been the cornerstone on which my masters degree has been built. Working under him has not only made me technically sound but has also made me more focused towards my vocation. I would like to thank Dr. Krzysztof J. Kochut for being considerate enough to be a part of my committee. His introductory lecture for the Enterprise Integration course was the first lecture I attended as a part of the masters program and was an enriching experience. I would like to thank Dr. Hamid R. Arabnia, not only for being a part of my committee but for also for the immense source of help and support he has been for me in these past two years. His course on Image Processing was an enjoyable and knowledgeable experience.

I would like to thank my family and friends for patiently bearing with me always. Their encouragement and faith in my abilities have been integral towards the completion of my degree.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the biggest challenges that computer scientists face today is to efficiently utilize the potential of the enormous amount of available data. Big Data, a term that came into being around 1997 [1] has exponentially gained popularity in the last decade. It has made its way into various aspects of our day to day lives. It has become an integral part of computer science research that ranges from complex topics like mapping of the human brain [2] to environmental research based on solar radiation [3]. Big Data can be better understood using the terms Size and Speed. Size, referring to the enormous amount of data present and Speed referring to the need for this data to be processed quickly to prove beneficial to us.

Recommender Systems is one of the popular topics associated with Big Data Analytics. Due to its huge number of applications in real life it is important that this area be actively researched and understood to reap potential benefits. The phenomenon of bringing products to users via the World Wide Web (www) has been revolutionary for the consumer based industry. Prime examples of companies that have hugely benefited from e-commerce are Amazon, eBay and Wal-Mart. As of March 2017, Amazon sites recorded about 183 million unique visitors per month in the United States of America[1]. Amazon further recorded a net revenue of 135.99 billion dollars through its e-commerce and service sales in 2016 [2]. Given the enormous amount of user traffic on these websites that has led to the success of these companies, other e-commerce organizations have also started focusing on improving the user interaction experience of their websites to boost their revenue. One of the key aspects of

[1]https://www.statista.com/statistics/271450/monthly-unique-visitors-to-us-retail-websites
[2]https://www.statista.com/statistics/266282/annual-net-revenue-of-amazoncom

Table 1.1: Popular recommender systems and their products

Recommender System	Products
Amazon.com	Varied Merchandise
Facebook	Friends, Advertisements, News
YouTube	Online Videos
Pandora	Music
Jester	Jokes
Netflix	Online TV shows and Movies
LinkedIn	Professional Connections
Google Search	Search related links
TripAdvisor	Travel Products

improving user experience on websites is by recommending the right products. Table 1.1 shows items recommended by some well-known real-world recommender systems [4].

A Recommender System refers to a technique or software that helps in providing suggestions about items that a user is most likely to prefer based on previous knowledge about the user or similar users [5]. Suggesting the right products to a user is an important part of increasing customer satisfaction and ensuring loyalty [6]. Recommender Systems can be considered as a part of Knowledge Discovery in Databases (KDD) techniques [7] and their development started in the early 1990s [8]. Extensive research that has been performed on recommender systems by popular e-commerce sites like Amazon, Pandora, Yahoo and Netflix has led to many proposed methods for generating recommendations [8–11]. A few familiar research prototypes are PHOAK, Syskills and Webert, Fab, and GroupLens.

One of the most successful approaches that is used to build a recommender system is by employing an approach known as Collaborative Filtering [7]. It is a technique used to predict a user's behavior towards an item based on that user's previous data or based on the information of other users that are associated with that item or items in similar domain. It works by assuming that if a group of people had a common preference in the

past they will share a common preference in the future [12]. There are several prediction methods that fall under the umbrella of Collaborative Filtering [11]. These methods can be broadly categorized as Memory-Based Collaborative Filtering Methods often known as Neighborhood-Based Methods and Model-Based Collaborative Filtering Methods [4].

In this document, we first discuss Memory-Based and Model-Based collaborative filtering techniques as well as the results of implementing these techniques as a package for recommender systems in ScalaTion [13]. The package uses the MovieLens [14] dataset for testing the performance of the implementations. The results have been evaluated using Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) as the accuracy metrics. The values for these metrics were obtained while performing 5-fold cross-validation on the predictions obtained during the experiments.

The primary contribution of this research is to shed light on the time-accuracy trade-offs that occur when different prediction methods are employed to perform recommendations. It allows us to make a calculated decision regarding the choice of prediction method while adopting a recommendation technique for our system. Apart from this, a timestamp analysis of the predictions has also been performed. The motivation behind this analysis is to find how accurately we can predict the rating for an item that the user is going to rate in the future.

The rest of this document is organized as follows: Section 2 further elucidates the importance of Recommender Systems and provides an insight on various models that have been implemented in our package to build an efficient recommender system. This section elaborates on Collaborative Filtering and discusses the existing recommender system building techniques that fall under it. It also tells us about some of the metrics that are used to evaluate recommender systems. Section 3 provides information regarding the dataset used and elaborates on the accuracy metrics used to determine the validity of our predictions. Section 4 discusses the results obtained while performing the experiments followed by Sec-

tion 5 where we talk about the inferences drawn from our experiments as well as the future of this research.

Chapter 2

Related Work

A facility that comprises a wide-ranging set of applications that deal with predicting user responses to choices is referred to as a recommender system [15]. The primary goal of a recommender system is to utilize available resources/data to understand customers preferences and provide them with suitable choices. Technically, the choices or recommendations are referred to as items and the customers are referred to as users. The basic underlying principle behind the working of a recommender system is that there exists a relation between a user and their choice of item. For example, if a user has an inclination towards a specific brand of clothes there is a high possibility that he will buy clothes of that brand in future. Thus, making it natural for a system to recommend products of that brand to the user. In this document, we are concentrating on real world scenarios where the user provides us with a numerical score referred to as a rating that tells us about his inclination towards an item. The score forms the relation between the user and the item and is utilized to predict his ratings for other items. Such scenarios can be modeled in the form of a user-item ratings matrix. The research presented in this document primarily focuses on the application of collaborative filtering techniques on such user-item-rating matrices to generate predictions.

2.0.1 Basic Models of Recommender Systems

Recommender System Models can be categorized based on the way data is available to them. Data can either be available because of user-item interaction which could be in the form of scores/ratings or in the form of meta-data about users or items in the form of keywords or

texts [10]. Based on data we can broadly classify recommender system models as Collaborative Filtering Models, Content-Based Models, Knowledge-Based Models, Demographic Models and Hybrid/Ensemble-Based Models [4]. In our research we focus on Collaborative Filtering Models. Equation 2.1 represents a user-item matrix A

$$A \in \mathbb{N}^{m*n} \tag{2.1}$$

where a_{ij} indicates the score by user i for item j; $a_{ij} = 0$ indicates an element that has not yet been scored. Our primary goal is to accurately predict these missing scores.

In order to perform a comparative evaluation, we have introduced a separate sub-category under Collaborative Filtering. This sub-category is referred to as Baseline Techniques

Baseline Techniques

This category consists of techniques that form a baseline for comparing other techniques. They are a group of techniques which predict values based on a few statistical operations that are logically related to the process of making a prediction. These techniques are as follows:

Random (RN) In this method we randomly predict a rating from a fixed range of values. In our experiments since we are dealing with movie ratings from 1-5 we predict the rating as a random number between 1-5.

Matrix Mean (MM) Predict the rating of an item j as the mean of all the ratings available.

$$r_{ij} = \frac{1}{n} \sum_{j=1}^{n} \mu(\mathbf{a}_{*j}^{+}) \tag{2.2}$$

where μ computes the mean of a vector, $+$ indicates a filtered column vector a_{*j} containing non-zero elements and j represents the column index.

User Mean (UM): Predict the rating of an item j as the mean of all the ratings provided by user i.

$$r_{ij} = \mu(\mathbf{a}_i^+) \tag{2.3}$$

Item Mean (IM) Predict the rating of item j as the mean of all the ratings that item j has received.

$$r_{ij} = \mu(\mathbf{a}_{*j}^+) \tag{2.4}$$

Mean of User and Item Average (AM) Predict the rating as the arithmetic mean (AM) of UM and IM.

$$r_{ij} = \mu(UM, IM) \tag{2.5}$$

In a similar way, we may define the Geometric Mean (GM) of UM and IM.

Weighted Average (WA) Predict the rating as the weighted aggregate of UM and IM. This technique is represented by Equation 2.6 where w is the normalized weight assigned and r_{ij} is the rating predicted for user i and item j.

$$r_{ij} = w \cdot UM + (1 - w) \cdot IM \tag{2.6}$$

Collaborative Filtering Models

These models are formulated when the available data has been obtained based on user-item interaction. Collaborative Filtering Models have been discussed in detail later in this section as majority of the techniques presented in this document fall under this Model.

2.0.2 Evaluating Recommender Systems

Since the development of Recommender Systems is essential due to its varied applications, it is necessary that we evaluate them correctly to get accurate results. The commonly used method for evaluation of recommender systems is the Offline Evaluation method. This

method and has been utilized to evaluate the techniques implemented as a part of our recommender package. [4].

Offline Evaluation using Historical Data

Offline methods are the most frequently used evaluation techniques for recommender system. In this method, we already have the final values at hand to which our predictions must be compared. One of the key ways to perform offline evaluation is by using Accuracy Metrics. A few commonly encountered accuracy metrics for recommender systems are Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Precision and Recall. The recommender systems developed as a part of the research discussed in this document have been evaluated using MAE and RMSE.

Chapter 3

Collaborative Filtering

An important aspect of e-commerce these days is the provision of gathering online reviews/ratings for products. Online user feedback, plays a key role in decision-making these days. One of the primary uses of user feedback is in the building of recommender systems using a technique called Collaborative Filtering [16]. Collaborative Filtering is a technique where we utilize user opinions to filter and evaluate items to perform recommendations. Although this technique may have gained popularity in the past couple of decades, it was prevalent before as what we commonly refer to as word of mouth. For example, if we receive good feedback about a movie or a product from many people there is a high possibility that we will watch that movie or buy that product. The Internet has made this word of mouth information available to us digitally so that our decisions are not limited to the opinions of a few individuals. In this section, we will be looking at techniques that fall under Collaborative Filtering and we have implemented them in our package. In general the recommendation process using collaborative filtering can be summarized by Figure 3.1

Memory Based Collaborative Filtering

Memory Based Collaborative Filtering techniques are based on the principle that similar users will have similar behavior towards items and similar items will receive identical preference from users. Based on this baseline principle Memory Based Collaborative techniques can be divided in two categories. User-based collaborative filtering where we predict the rating of an item by a user based on his similarity with other users and Item-based collaborative filtering where we predict the rating of a target item based on the its similarity

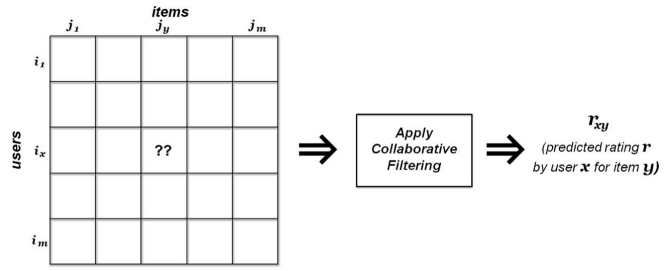

Figure 3.1: Recommendation process using collaborative filtering

with other items. This requires finding similarity between entities. We have utilized the following measures in our implementations to find similar items and users. These measures are Cosine-based Similarity, Correlation-based Similarity and Adjusted Item-Cosine Similarity [17]. Since the process involves finding similar items and users we need to find a condition under which we can compare two users or items and then find their similarity coefficient. In our experiments we find the similarity coefficients between two items if they have been rated by the same users. Similarly, we find similarity coefficients between two users if they have rated the same items.

Equation 3.1 defines a set of items that have been rated by users x *and* y

$$I_{xy} = \{l \mid a_{xl}a_{yl} > 0\} \tag{3.1}$$

Equation 3.2 defines a set of users that have rated items x *and* y

$$U_{xy} = \{l \mid a_{lx}a_{ly} > 0\} \tag{3.2}$$

Cosine-based Similarity In this case two users or items are represented as vectors. The similarity between them is obtained by calculating the cosine of the angle between the two vectors [17]. Equation 3.4 represents calculation of cosine similarity between two users x and

y.

$$usim(x, y) = cos(\mathbf{a}_x, \mathbf{a}_y) \tag{3.3}$$

$$cos(\mathbf{a}_x, \mathbf{a}_y) = \frac{\sum\limits_{l \in I_{xy}} a_{xl} a_{yl}}{\sqrt{\sum\limits_{l \in I_{xy}} (a_{xl})^2} \sqrt{\sum\limits_{l \in I_{xy}} (a_{yl})^2}} \tag{3.4}$$

Equation 3.6 defines a function $sim(x, y)$ to show how similarity between two items x and y is calculated.

$$sim(x, y) = cos(\mathbf{a}_{*x}, \mathbf{a}_{*y}) \tag{3.5}$$

$$cos(\mathbf{a}_{*x}, \mathbf{a}_{*y}) = \frac{\sum\limits_{l \in U_{xy}} a_{lx} a_{ly}}{\sqrt{\sum\limits_{l \in U_{xy}} (a_{lx})^2} \sqrt{\sum\limits_{l \in U_{xy}} (a_{ly})^2}} \tag{3.6}$$

The code snippet below shows the implementation of the function $cos(x : MatrixD)$ in ScalaTion that calculates the cosine similarity based on the columns of an input matrix x.

```
def cos (x: MatrixD): MatrixD =
{
    val cs = eye (x.dim2)
    val xtx = x.t * x
    val nx = for (j <- cs.range2) yield
                 x.col(j).norm
    for (i <- cs.range1; j <- 0 until i) {
        cs(i, j) = xtx(i, j) / nx(i) * nx(j)
        cs(j, i) = cs(i, j)
    } // for
    cs
} // cos
```

Correlation-based Similarity: By calculating Pearson's correlation coefficient we can determine if two items or two users represented by x and y are similar to each other [17]. Equation 3.8 represents how similarity between items is calculated.

$$sim(x, y) = corr(\mathbf{a}_{*x}, \mathbf{a}_{*y}) \tag{3.7}$$

$$corr(\mathbf{a}_{*x}, \mathbf{a}_{*y}) = cos(\mathbf{a}_{*x} - \mu(\mathbf{a}^{+}_{*x}), \mathbf{a}_{*y} - \mu(\mathbf{a}^{+}_{*y})) \tag{3.8}$$

The similarity function $usim(x, y)$ is related to finding similarity between users and can be represented by Equation 3.10.

$$usim(x, y) = corr(\mathbf{a}_{x}, \mathbf{a}_{y}) \tag{3.9}$$

$$corr(\mathbf{a}_{x}, \mathbf{a}_{y}) = cos(\mathbf{a}_{x} - \mu(\mathbf{a}^{+}_{x}), \mathbf{a}_{y} - \mu(\mathbf{a}^{+}_{y})) \tag{3.10}$$

Adjusted Cosine-based Similarity: To address the drawbacks obtained from pure cosine and pure correlation based similarity, adjusted cosine similarity [17] was introduced. Adjusted Cosine similarity differs from the above two similarity measures in that adjusted cosine similarity subtracts the corresponding mean user rating while calculating the similarity between items. Equation 3.11 represents calculation of a set of users S where each user has rated both item x and y.

$$S_{xy} = \{l | a_{lx} a_{ly} > 0\} \tag{3.11}$$

The similarity function $sim(x, y)$ to calculate the adjusted cosine similarity is given by Equation 3.12.

$$sim(x, y) = \frac{\sum\limits_{l \in S_{xy}} (a_{l,x} - \mu(\mathbf{a}^{+}_{l}))(a_{l,y} - \mu(\mathbf{a}^{+}_{l}))}{\sqrt{\sum\limits_{l \in S_{xy}} (a_{l,x} - \mu(\mathbf{a}^{+}_{l}))^2} \sqrt{\sum\limits_{l \in S_{xy}} (a_{l,y} - \mu(\mathbf{a}^{+}_{l}))^2}} \tag{3.12}$$

where $\mu(\mathbf{a}^{+}_{\mathbf{l}}))$ represents the average rating for each user l that has rated both items x and y.

Predictions: Once the similarity between entities have been calculated we can predict a rating r denoted by $r_{i,j}$ for a user i and item j. This value is calculated by Equation 3.13 for item-based similarities and by Equation.

$$r_{ij} = \frac{\sum_{k=1}^{n}(sim(j,k) \cdot a_{ik})}{\sum_{k=1}^{n}(|sim(j,k)|)} \; where \; a_{ik} \neq 0 \tag{3.13}$$

Prediction of missing ratings using user-based similarity values is done by Equation 3.14

$$r_{ij} = \frac{\sum_{k=1}^{m}(usim(i,k) \cdot a_{kj})}{\sum_{k=1}^{m}(|usim(i,k)|)} \; where \; a_{kj} \neq 0 \tag{3.14}$$

Model Based Collaborative Filtering

Model based Collaborative Filtering uses machine learning and data mining techniques to create mathematical predictive models which help us to generate predictions. Thus, model based collaborative filtering contains two parts, the training or the model building phase and the prediction phase. Our package utilizes Singular Value Decomposition as a model based technique to generate predictions. Model based techniques are prone to Overfitting. Overfitting issues have been addressed in our experiments and are explained in detail in the following sub-sections.

Singular Value Decomposition (SVD) Singular Value Decomposition coupled with Dimensionality reduction is one of the common model based techniques employed to generate accurate predictions [7]. SVD is a method of decomposing a real matrix as into three separate matrices such that the product of the three matrices is equal to the original matrix. Mathematically, Singular Value Decomposition can be represented by Equation 3.15

$$SVD(A) = USV^t \tag{3.15}$$

where A is a real $m \times n$ matrix and $m >= n$, U is an $m \times n$ matrix of orthogonal eigen vectors of $A \cdot A^t$ (left singular vectors). S is an $n \times n$ diagonal matrix containing square roots

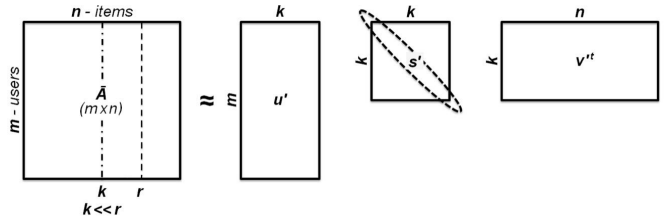

Figure 3.2: Dimensionality Reduction in SVD

of the eigenvalues of $A^t \cdot A$ and $A \cdot A^t$ (singular values). V is an $n \times n$ matrix of orthogonal eigen vectors of $A^t \cdot A$ (right singular vectors). We have implemented the Singular Value Decomposition algorithm as a part of the ScalaTion framework. The 'SVD' class implemented in ScalaTion is used to compute the SVD of matrix 'A' using the Golub-Kahan-Reinsch Algorithm [18]. It involves transforming the input matrix to its Bidiagonal form using the Householder Transformation [19] procedure. The Bidiagonal matrix is then used to obtain singular values using the QR-Factorization algorithm [20].

Generating Predictions: We utilize dimensionality reduction as a technique to simplify the generation of predictions using SVD [7]. *Dimensionality Reduction* is the method of reducing the number of dimensions of a dataset while ensuring that the original data is minimally affected. Dimensionality reduction used on top of SVD addresses the problem of Space Complexity that is associated with Memory-Based Collaborative Filtering Methods. This idea leads to obtaining a low-rank approximation of the original matrix [21]. This low-rank approximation can be elaborated by Equation 3.16 and Figure 3.2 below.

$$A^{'} \approx U^{'} S^{'} V^{'t} \tag{3.16}$$

Here, $U^{'} = m \times k$, $S^{'} = k \times k$ and $V^{'t} = k \times n$. Where k is a predefined dimension and $k << r$. r is the rank of the matrix. This low-rank approximation satisfies the *Fobrenius Normal* form, i.e. $||A - A^{'}||$ is as minimum as possible.

Prediction of the missing values in a matrix is performed in the following way [7].

1. Decompose the original matrix A using SVD to obtain the U, S and V.

2. Reduce the dimensions of these matrices to a dimension k decided initially to obtain $U^{'}, S^{'}$and $V^{'t}$.

3. Compute the square-root of the singular values in $S^{'}$.

4. Obtain matrices P by $U_k \cdot \sqrt{S_k}$ and Q by $\sqrt{S_k} \cdot V^t$.

5. Once P and Q have been obtained, we can predict the rating r_{ij} for a given user i for an item j by obtaining the dot product of the row vector $\mathbf{p_i}$ and column vector $\mathbf{q_{*j}}$. This can be represented by Equation 3.17 where $(\cdot)$ represents the dot product of the two vectors.

$$predict(i, j) = r_{ij} = \mathbf{p_i} \cdot \mathbf{q_{*j}} \tag{3.17}$$

Incremental Singular Value Decomposition (SVD) Simon Funk [22] introduced an incremental way of generating predictions by obtaining low rank approximate matrices using singular value decomposition coupled with gradient descent and regularization [23]. Gradient Descent serves as an error minimization technique while Regularization helps us to avoid problems we may encounter due to Overfitting. The algorithm works in the following way.

1. A reduced dimension k is chosen such that we have k feature vectors in the U and V matrices which are initialized with an extremely small value.

2. Gradient descent and regularization are employed as to reduce the error for each feature vector.

The minimization objective function E is represented using Equation 3.18 where F represents the Frobenius Normal form.

$$E = \min_{U,V} ||A - UV^t||_F^2 \; s.t. \; U \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{m \times k} \tag{3.18}$$

The entire U and V matrix is initially assigned with an extremely small value which is calculated according to Equation 3.20 where *nzmean* refers to the mean of the original matrix excluding the zeroes as represented by Equation 3.19.

$$nzmean = \frac{1}{n}\sum_{j=1}^{n}\mu(\mathbf{a}_{*j}^{+}) \tag{3.19}$$

$$u_{i,j} = v_{i,j} = \sqrt{\frac{nzmean}{k}} \tag{3.20}$$

Gradient Descent and Regularization are used together in the following way. We iterate over each feature vector while continuously updating the values in the U and V matrices. This process is repeated till the average sum of squared errors obtained from the difference of the original rating and the predicted rating is minimum for each feature vector. The error is calculated using Equation 3.21 whereas the value of $predict(i, j)$ is obtained using Equation 3.27.

$$err = (a_{ij} - predict(i, j)) \tag{3.21}$$

During the iterations the U and the V matrices are updated using the following equations.

$$userval = u_{ix} \tag{3.22}$$

$$itemval = v_{jx} \tag{3.23}$$

$$u_{ix} += \lambda \cdot (err \cdot itemval - \gamma \cdot userval) \tag{3.24}$$

$$v_{xj} += \lambda \cdot (err \cdot userval - \gamma \cdot itemval) \tag{3.25}$$

where λ is the learning rate, γ is the regularization constant and $x \in \{i, k\}$ is the the feature vector for user i and item j [22].

The final minimization objective function E can be represented using Equation 3.26

$$E = \min_{U,V} ||A - UV^t||_F^2 + \lambda(||U||_F^2 + ||V||_F^2) \tag{3.26}$$

Post minimization the rating r_{ij} for a user i and item j is given by Equation 3.27 where $(\cdot)$ represents the dot product of the two row vectors $\mathbf{u_i}$ and $\mathbf{v_j}$.

$$predict(i, j) = r_{ij} = \mathbf{u_i} \cdot \mathbf{v_j} \tag{3.27}$$

Challenges faced by Collaborative Filtering Techniques

In this subsection we mention some of the common challenges faced by collaborative filtering techniques and the steps we have taken during our implementations to address them [11].

Sparsity: As our goal is to solve real world problems which involves dealing with large datasets, it is usually difficult to obtain data that is not sparse in nature. Consider a large user-item ratings dataset from a product based website. It is unlikely that each user will have rated atleast 50% of all the products available. This makes it very difficult to find co-related users and items. The dataset that we have used for our experiments is 93.6953% sparse and highlights this issue. Sparsity has been addressed in our implementation by performing column-mean imputation [7]. Although, [7] applies column-mean imputation only for predictions obtained using SVD we have used it in our experiments with Memory-Based methods too.

Shilling Attacks: Shilling attacks, refers to scenarios when people provide a large number of high ratings to their own product or extremely low ratings to products of their competitors to boost the sales of their products. This problem can be overcome by using normalization techniques as suggested by Bell and Koren [24]. As a part of our recommender package we have implemented two normalization techniques, namely z-scores and subtraction of item-mean from the respective item ratings in the input matrix. As mentioned in [7] normalization performed by item-mean subtraction yields better results compared to z-scores.

Space Complexity: Memory based recommender systems have high space complexities that increases with the increase in the number of users and items. User based methods will have

$O(m^2)$ and item based methods will have $O(n^2)$ complexities considering an $m \times n$ user-item matrix. Usually model based methods generate training models that have lower space complexities when compared to memory based methods.

Synonymy: Synonymy refers to the scenarios where similar items maybe named in different ways. For example, a notepad and a letter pad maybe be perceived as different items but are the same stationery.

To address the challenges of Space Complexity and Synonymy, model-based collaborative filtering techniques have been included in our package.

Chapter 4

Dataset and Evaluation

There are several public datasets that are available for performing research related to recommender systems. A few of them have been mentioned in the table 4.1.

Experiments were conducted using the MovieLens dataset (ML-100k) [14]. The MovieLens dataset was collected by the GroupLens Research Project at the University of Minnesota. This data set consists of 100,000 ratings (1-5) by 943 users for 1682 movies where each user has rated atleast 20 movies. The files that were used for conducting the experiments are as follows:

1. u.data: The full data set, 100000 ratings by 943 users on 1682 movies. Users and movies are numbered consecutively from 1. The data is randomly ordered and represented as

Table 4.1: Popular publicly available datasets

Dataset Name	Description
Amazon	merchandize ratings/reviews
MovieLens	user-item movie ratings
Yahoo!	movies, music and image ratings
Netflix Prize	movie ratings
Jester	joke ratings
Last.fm	music and radio station ratings
Yelp	restaurant reviews
Book-Crossing	book review ratings

a tab separated list where the timestamps are unix seconds since 1/1/1970 UTC
user id | item id | rating | timestamp

2. sorted-data.txt: Created by us by sorting u.data in non-ascending order based on the timestamp field.

3. u(1-5).base - u(1-5).test: The data sets u(1-5).base and u(1-5).test have been obtained by splitting u.data in the ratio 80:20 to serve as training and test data respectively.

Evaluation: All the results obtained during the experiment have been evaluated on the basis of Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) as the accuracy metrics. These values were obtained by performing 5-fold cross-validation of the predicted ratings. Cross-validation works in the following way. For Phase 1 of our experiments u(1-5).base obtained from u.data becomes our training dataset. We predict the values for the users and items that are present the corresponding u(1-5).test dataset. The predictions are validated with the rating that are provided to us in the test dataset. This is repeated for all the splits provided to us. In Phase 2 we divide the entire u.data file into 5 parts (5-fold) we choose Part 1 as the test data set and the remaining as the training dataset. We obtain the predictions based on the training dataset and cross-validate the predictions with the values in the test dataset. This process is repeated till each of the 5 parts has served as the testing dataset once. MAE and RMSE for the experiments are represented by Equation 4.1 and Equation 4.2 respectively.

$$MAE = \frac{\sum_{i,j} |a_{ij} - r_{ij}|}{n} \tag{4.1}$$

$$RMSE = \sqrt{\frac{E}{n}} \tag{4.2}$$

where a_{ij} represents the original rating given by a user i, for item j and r_{ij} represents the corresponding predicted value and E represents the objective function from 3.18

Chapter 5

Experiments and Results

The experiments have been performed in three phases. In each phase the results have been divided into three categories, Baseline Predictors, Memory-Based Collaborative Filtering Methods and Model-based Collaborated Filtering Methods.

5.0.1 Phase 1

We tested our implementation with the splits (u(1-5).base - u(1-5).test) provided in the MovieLens dataset as mentioned before. This is done to verify the accuracy of our predictions against pre-established results. As our primary goal is to explore the time-accuracy trade-offs in recommender systems, we first established the desired accuracy for each technique. Once the correct accuracy was obtained we proceeded to the next step of obtaining the runtime corresponding to these techniques. The tables and the figures below elucidate our results.
Baseline Predictors: Table 5.1 shows the performance of the techniques that fall under this category. A comparison of these techniques on the basis of accuracy and runtime can be observed in Figure 5.1 and Figure 5.2 respectively.

Memory Based Predictors: Table 5.2 shows the performance of the techniques that fall under this category. In this table n refers to *Normalization*. A comparison of these techniques on the basis of accuracy and runtime can be observed in Figure 5.3 and Figure 5.4 respectively.

Model Based Predictors: Table 5.3 shows the performance of the techniques that fall under this category. A comparison of these techniques on the basis of accuracy and runtime can be observed in Figure 5.5 and Figure 5.6 respectively.

Table 5.1: Phase 1: Performance of Baseline Predictors

Technique	MAE	MAE(r)	RMSE	RMSE(r)	Time(ms)
RN	1.50997	1.49940	1.88238	1.88238	8.96237
MM	0.93286	0.89416	1.12709	1.21392	24.31601
UM	0.83618	0.80488	1.04372	1.08379	728.94860
IM	0.81883	0.78602	1.02784	1.06432	1117.90673
AM	0.79503	0.74218	0.94585	0.95750	2233.97358
GM	0.79707	0.74414	0.98621	1.01309	2314.21008
WA (w = 0.55)	0.79410	0.74245	0.98275	1.01086	2392.17673

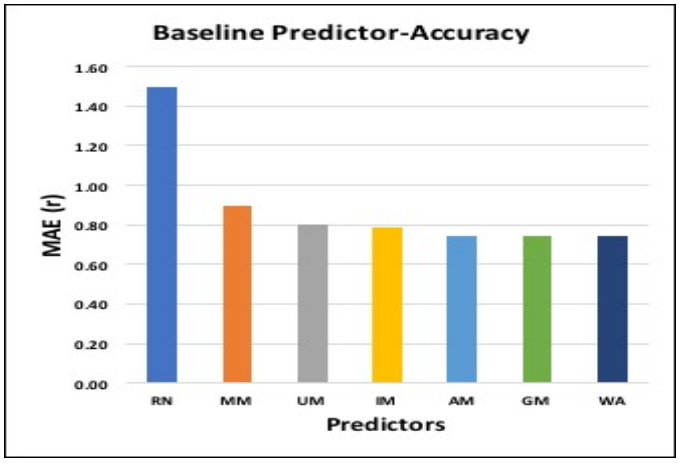

Figure 5.1: Predictor-Accuracy Graph for Baseline Predictors

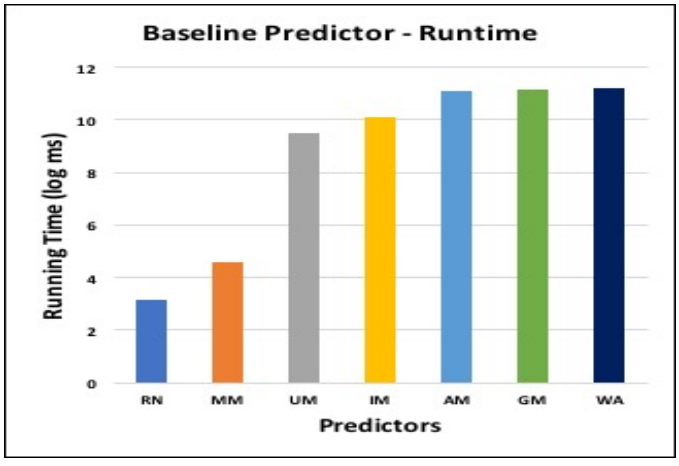

Figure 5.2: Predictor-Runtime Graph for Baseline Predictors

Table 5.2: Phase 1: Performance of Memory Based Predictors

Technique	MAE	MAE(r)	RMSE	RMSE(r)	Time(ms)
ucorr	0.81070	0.77815	1.02263	1.06051	4023.76185
nucorr	0.75252	0.71410	0.96782	1.01038	4852.08538
icorr	0.82225	0.79194	1.04466	1.08542	3957.94614
nicorr	0.76260	0.72848	0.97466	1.01679	6212.84473
mncorr	0.73942	0.70145	0.94592	0.98832	5099.41162
ucos	0.81368	0.78199	1.02558	1.06334	2376.73377
nucos	0.75259	0.71417	0.96800	1.01054	5852.61308
icos	0.85014	0.81782	1.08171	1.11694	3493.45730
nicos	0.75281	0.71906	0.96430	1.00692	6538.35095
mncos	0.73668	0.69776	0.94368	0.98548	6268.04633
acos	0.76212	0.72716	0.96703	1.00823	4765.78880

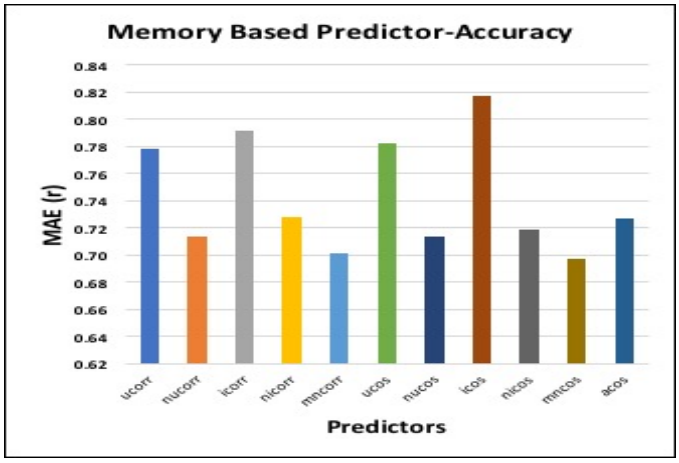

Figure 5.3: Predictor-Accuracy graph for Memory Based Predictors

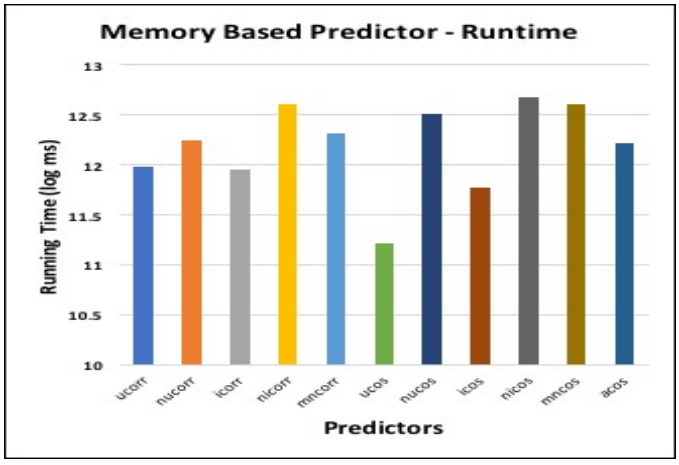

Figure 5.4: Predictor-Runtime graph for Memory Based Predictors

Table 5.3: Phase 1: Performance of Model Based Predictors

Technique	MAE	MAE(r)	RMSE	RMSE(r)	Time(ms)
svd (k =10)	0.77902	0.74549	0.98326	1.02397	544871.81870
svd (k =14)	0.77848	0.74487	0.98302	1.02423	623788.14210
svdreg (k =10)	0.75312	0.71571	0.95498	0.99725	16973.05425
svdreg (k =25)	0.73614	0.69852	0.93364	0.97741	54993.08445

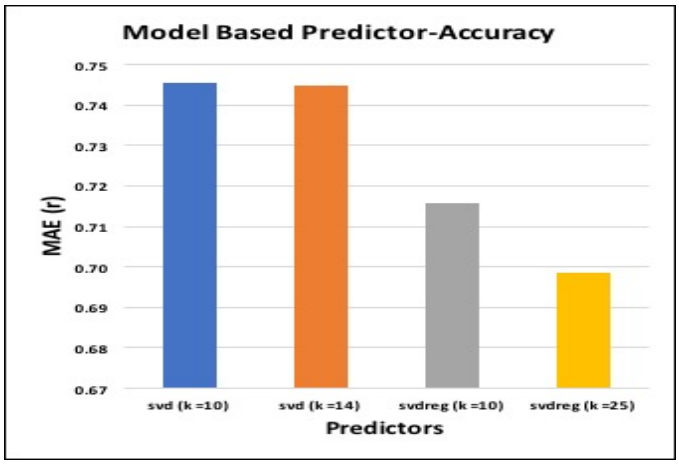

Figure 5.5: Predictor-Accuracy graph for Model Based Predictors

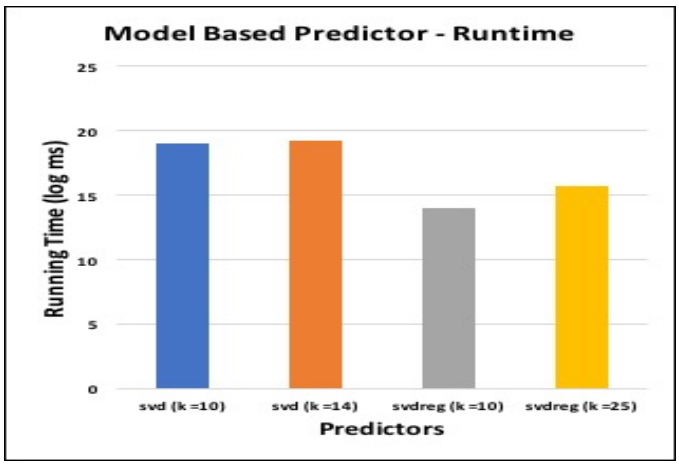

Figure 5.6: Predictor-Runtime graph for Model Based Predictors

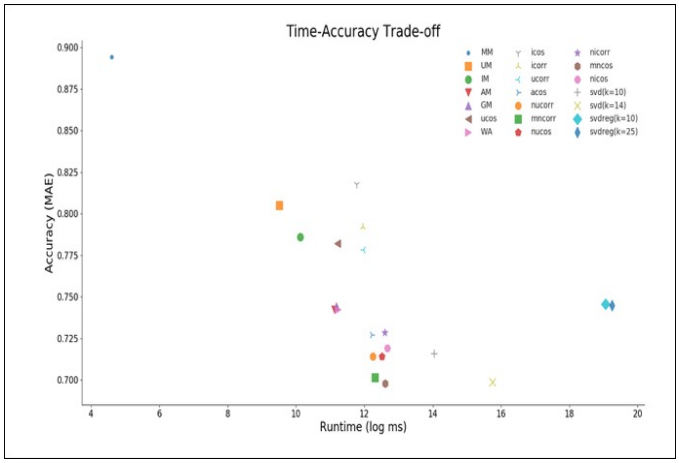

Figure 5.7: Time-Accuracy Trade-off graph for all prediction techniques

The Time-Accuracy Trade-Off analysis can be done using the Time-Accuracy Graph as shown in Figure 5.7.

We also compared our results on the basis of MAE for the aformentioned dataset with MyMediaLite [25], a well cited free recommender systems library[1]. Table 5.4 shows a comparsion between our package and MyMediaLite on the basis of MAE values. The values mentioned are those that were obtained after employing normalization as the values reported for MyMediaLite are also post normalized values.

5.0.2 Phase 2

In this phase we generate training and testing datasets in the ratio 80:20 respectively by randomly permuting the entries in u.data. Once these datasets have been obtained we execute the implemented collaborative filtering algorithms in Phase 1 to obtain and validate the results. There are two reasons behind implementing Phase 2. One, to ensure if the accuracy pattern observed in Phase 1 is not limited to the dataset splits provided to us by default. Two, we observed in Phase 1 that the mean of user and item cosine similarity yielded

[1]http://www.mymedialite.net/examples/datasets.html

Table 5.4: Comparison with MyMediaLite w.r.t MAE(r)

Technique	MyMediaLite	ScalaTion
Baseline	0.74503	0.74218
ucorr	0.72805	0.71410
icorr	0.71440	0.72848
ucos	0.73700	0.71417
icos	0.72700	0.71906
svd	0.72520	0.74487
svdreg	0.71944	0.69852

Table 5.5: Phase 2: Performance of Baseline Predictors

Technique	MAE	MAE(r)	RMSE	RMSE(r)
RN	1.50725	1.507255	1.88644	1.88644
MM	0.93810	0.89416	1.12739	1.21990
UM	0.83618	0.80488	1.04372	1.08379
IM	0.81883	0.78602	1.02784	1.06432
AM	0.80755	0.75886	1.00342	1.03265
GM	0.80286	0.75576	0.99420	1.02558
WA (w = 0.55)	0.80804	0.76204	1.00564	1.03742

more accurate results when compared to the accuracy obtained when these methods were individually implemented.

Baseline Predictors: Table 5.5 shows the performance of the techniques that fall under this category.

Memory Based Predictors: Table 5.6 shows the performance of the techniques that fall under this category.

Model Based Predictors: Table 5.7 shows the performance of the techniques that fall under this category.

Table 5.6: Phase 2: Performance of Memory Based Predictors

Technique	MAE	MAE(r)	RMSE	RMSE(r)
ucorr	0.8854176	0.8560881	1.1454856	1.1807995
nucorr	0.8761564	0.8423802	1.1442285	1.1788608
icorr	0.9140582	0.8880070	1.184166	1.2209844
nicorr	0.9136171	0.885971	1.1910061	1.2277214
mncorr	0.8554967	0.8223405	1.1055369	1.1420012
ucos	0.8893268	0.8599228	1.1479589	1.1844477
nucos	0.8760753	0.8422496	1.1595316	1.1928800
icos	0.9251769	0.8958652	1.203954	1.2392736
nicos	0.9148547	0.8886451	1.200916	1.2127386
mncos	0.8353354	0.800895	1.0785917	1.1159729
acos	0.9074930	0.8821422	1.197698	1.225921

Table 5.7: Phase 2: Performance of Model Based Predictors

Technique	MAE	MAE(r)	RMSE	RMSE(r)
svd (k =10)	0.831858	0.799188	1.04862	1.08287
svd (k =14)	0.829341	0.799038	1.04714	1.08213
svdreg (k =10)	0.783969	0.7481075	0.992672	1.0329455
svdreg (k =25)	0.7821163	0.73765	0.9875430	1.0389541

Table 5.8: Phase 3: Performance of Baseline Predictors

Technique	MAE	MAE(r)	RMSE	RMSE(r)	Success
RN	1.514	1.514	1.88898	1.888988	19.964
MM	0.9489213	0.91092	1.13071	1.222405	32.892
UM	0.927425	0.892955	1.161291	1.183338	33.33321
IM	0.832758	0.800277	1.038046	1.07352	36.02233
AM	0.8295275	0.7927499	1.0338455	1.0663542	36.38825
GM	0.8290106	0.7925095	1.0331032	1.06582	36.568077
WA	0.8281514	0.7909494	1.0316421	1.064200	36.464178

5.0.3 Phase 3

We present a novel experiment where we try to perform a timestamp analysis of the accuracy obtained by employing techniques presented in the previous sections. In this analysis we sort u.data according to the timestamp field in non-ascending order. We retain the first 7500 ratings and replace the rest by zero. We then try to predict these unknown ratings using the techniques verified in the previous phases. These predictions are then cross-validated with the original values in in u.data. We repeat this process by continuously increasing the number of known ratings by 50 and predicting the rest of the values in each iteration. The accuracy of the overall predictions for a particular algorithm is then evaluated using MAE and RMSE. The idea here is to observe if we can accurately predict the rating for a movie that a user is going to rate in the future. The column labeled as "Success" in the Predictor tables below, displays the success percentage corresponding to each technique. Success is when the rounded predicted rating and the corresponding rating in the test dataset are exactly the same.

Baseline Predictors: Table 5.8 shows the accuracy and the success percentage of the techniques that fall under this category.

Table 5.9: Phase 3: Performance of Memory Based Predictors

Technique	MAE	MAE(r)	RMSE	RMSE(r)	Success
ucorr	0.871869	0.837973	1.101765	1.137582	36.423644
nucorr	0.856695	0.824711	1.095781	1.132344	37.275774
icorr	0.979145	0.956301	1.252936	1.286809	32.586468
nicorr	0.863040	0.830656	1.102499	1.138054	36.739241
mncorr	0.831974	0.806054	1.061105	1.103252	37.332300
ucos	0.877239	0.843575	1.106344	1.143774	36.196025
nucos	0.8566951	0.82471098	1.0957810	1.1323436	37.275774
icos	1.001486	0.978266	1.275042	1.314210	32.526601
nicos	0.8537831	0.8199012	1.0901771	1.1243605	37.07363536
mncos	0.829569	0.804096	1.057799	1.101268	37.481908
acos	0.911645	0.869573	1.158297	1.177301	35.220492

Table 5.10: Phase 3: Performance of Model Based Predictors

Technique	MAE	MAE(r)	RMSE	RMSE(r)	Success
SVD (k = 10)	0.8379374	0.80828	1.0834545	1.119268	35.151934
SVD (k =14)	0.838703781	0.80884	1.05344	1.090898	35.908
svdreg (k = 5)	0.8157233	0.78392	1.0277241	1.0646814	37.36
svdreg (k = 10)	0.840322009	0.79472	1.026412	1.06492	37.188999

Memory Based Predictors: Table 5.9 shows the accuracy and the success percentage of the techniques that fall under this category.

Model Based Predictors: Table 5.10 shows the accuracy and the success percentage of the techniques that fall under this category.

Chapter 6

Conclusion and Future Work

Collaborative Filtering techniques were implemented and added to an open source big data analytics framework called ScalaTion. This implementation has allowed us to make the following inferences. From the Time-Accuracy study of prediction models implemented in ScalaTion we infer that depending on our priority we now have a choice of technique while building recommender systems. In cases where time/speed has a higher priority than accuracy and storage, Memory Based Collaborative Filtering techniques are preferred. In situations where accuracy and storage take precedence we prefer Model Based Collaborative Filtering techniques.

An important contribution is the implementation of the weighted aggregate method which provides us with a baseline technique that is time and storage effective and performs nearly as well as some of the Collaborative Filtering methods. One of the other contributions of this paper is the application of Normalization to Memory Based Collaborative Filtering techniques. We observe that despite being a costly procedure with respect to time, Normalization can significantly improve the accuracy our predictions. Apart from guaranteeing the correctness of our implementation, one of the important observations from Phase 1 of our experiments is that the mean of the final ratings obtained from the combined predictions of user cosine similarity and item cosine similarity is more accurate than employing these techniques individually. Although this increases the overall runtime of the process we see from the results that there is a very noticeable improvement in the overall accuracy. This deduction is confirmed from the results obtained in Phase 2 of our experiments. From the prediction results obtained during the timestamp analysis conducted in Phase 3 of our experiments we

are able to deduce the accuracy with which we can predict the rating for a movie that a user will watch in the future. From the results obtained in Phase 3 of our experiments we infer that Memory Based Collaborative Filtering techniques must be preferred over Model Based techniques when predicting timestamp based future ratings.

Moving forward we plan to include more methods of generating recommendations. Some of the other techniques that can be implemented in future are the Sigmoid Asymmetric Factor Model [26], Bayesian belief nets [27] and MDP-based Collaborative Filtering methods [28]. An folding-in techqniue [29] using incremental singular value decomposition is another well-known method that can be adopted within our package. Complex Hybrid Recommenders can also be implemented to yield better results. Since parallel computation is a faster method of performing executions we can try to make these implementations parallel in order to improve the performance of some of these algorithms. The ScalaTion *par* directory [30] gives a detailed explanation as to how linear algebra operations can be implemented in parallel to give better runtime performance.

Bibliography

[1] J. R. Mashey, "Big data and the next wave of infras-tress," in *Computer Science Division Seminar, University of California, Berkeley*, 1997.

[2] G. Lohmann, D. S. Margulies, A. Horstmann, B. Pleger, J. Lepsien, D. Goldhahn, H. Schloegl, M. Stumvoll, A. Villringer, and R. Turner, "Eigenvector centrality mapping for analyzing connectivity patterns in fmri data of the human brain," *PloS one*, vol. 5, no. 4, p. e10232, 2010.

[3] O. Şenkal and T. Kuleli, "Estimation of solar radiation over turkey using artificial neural network and satellite data," *Applied Energy*, vol. 86, no. 7, pp. 1222–1228, 2009.

[4] C. C. Aggarwal, *Recommender Systems: The Textbook*, 1st ed. Springer Publishing Company, Incorporated, 2016.

[5] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, *Recommender Systems Handbook*, 1st ed. New York, NY, USA: Springer-Verlag New York, Inc., 2010.

[6] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '08. New York, NY, USA: ACM, 2008, pp. 426–434. [Online]. Available: http://doi.acm.org/10.1145/1401890.1401944

[7] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl, "Application of dimensionality reduction in recommender system – a case study," in *IN ACM WEBKDD WORKSHOP*, 2000.

[8] J. Bennett, S. Lanning, and N. Netflix, "The netflix prize," in *In KDD Cup and Workshop in conjunction with KDD*, 2007.

[9] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. on Knowl. and Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005. [Online]. Available: https://doi.org/10.1109/TKDE.2005.99

[10] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009. [Online]. Available: http://dx.doi.org/10.1109/MC.2009.263

[11] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Adv. in Artif. Intell.*, vol. 2009, pp. 4:2–4:2, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1155/2009/421425

[12] C.-C. Ma, "A guide to singular value decomposition for collaborative filtering," 10 2017.

[13] V. G. Harish, V. K. Bingi, and J. A. Miller, "A big data platform integrating compressed linear algebra with columnar databases," in *2016 IEEE International Conference on Big Data (Big Data)*, Dec 2016, pp. 2344–2352.

[14] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 5, no. 4, p. 19, 2016.

[15] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of Massive Datasets*, 2nd ed. New York, NY, USA: Cambridge University Press, 2014.

[16] M. D. Ekstrand, J. T. Riedl, J. A. Konstan *et al.*, "Collaborative filtering recommender systems," *Foundations and Trends® in Human–Computer Interaction*, vol. 4, no. 2, pp. 81–173, 2011.

[17] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 285–295.

[18] P. A. Businger and G. H. Golub, "Algorithm 358: Singular value decomposition of a complex matrix [f1, 4, 5]," *Commun. ACM*, vol. 12, no. 10, pp. 564–565, Oct. 1969. [Online]. Available: http://doi.acm.org/10.1145/363235.363249

[19] G. H. Golub and C. F. Van Loan, *Matrix Computations (3rd Ed.)*. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[20] B. N. Parlett, "The qr algorithm," *Computing in Science and Engg.*, vol. 2, no. 1, pp. 38–42, Jan. 2000. [Online]. Available: http://dx.doi.org/10.1109/5992.814656

[21] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, p. 391, 1990.

[22] G. Piatetsky, "Interview with simon funk," *SIGKDD Explor. Newsl.*, vol. 9, no. 1, pp. 38–40, Jun. 2007. [Online]. Available: http://doi.acm.org/10.1145/1294301.1294311

[23] M. Percy, "Collaborative filtering for netflix," 2009.

[24] R. M. Bell and Y. Koren, "Improved neighborhood-based collaborative filtering," in *KDD cup and workshop at the 13th ACM SIGKDD international conference on knowledge discovery and data mining*. sn, 2007, pp. 7–14.

[25] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Mymedialite: A free recommender system library," in *Proceedings of the Fifth ACM Conference on Recommender Systems*, ser. RecSys '11. New York, NY, USA: ACM, 2011, pp. 305–308. [Online]. Available: http://doi.acm.org/10.1145/2043932.2043989

[26] R. M. Bell, Y. Koren, and C. Volinsky, "The bellkor 2008 solution to the netflix prize," *Statistics Research Department at AT&T Research*, 2008.

[27] X. Su and T. M. Khoshgoftaar, "Collaborative filtering for multi-class data using belief nets algorithms," in *Tools with Artificial Intelligence, 2006. ICTAI'06. 18th IEEE International Conference on.* IEEE, 2006, pp. 497–504.

[28] G. Shani, D. Heckerman, and R. I. Brafman, "An mdp-based recommender system," *Journal of Machine Learning Research*, vol. 6, no. Sep, pp. 1265–1295, 2005.

[29] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Incremental singular value decomposition algorithms for highly scalable recommender systems," in *Fifth International Conference on Computer and Information Science*, 2002, pp. 27–28.

[30] Y. L. LI, "Evaluation of parallel implementation of dense and sparse matrix for sclation library."