EFFICIENT RATING ESTIMATION BY USING SIMILARITY IN MULTI-DIMENSIONAL CHECK-IN DATA

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

BEHLÜL UÇAR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2014

Approval of the thesis:

EFFICIENT RATING ESTIMATION BY USING SIMILARITY IN MULTI-DIMENSIONAL CHECK-IN DATA

submitted by **BEHLÜL UÇAR** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

ABSTRACT

EFFICIENT RATING ESTIMATION BY USING SIMILARITY IN MULTI-DIMENSIONAL CHECK-IN DATA

Uçar, Behlül
M.S., Department of Computer Engineering
Supervisor : Assoc. Prof. Dr. Pınar Karagöz
Co-Supervisor : Prof. Dr. İ. Hakkı Toroslu

September 2014, 64 pages

The usage coverage of location based social networks have boomed in the last years as well as the amount of data produced in them. This data is suitable for processing in order to make prediction. One of the requirements of this process is that the method used should be suitable for very big data sets.

We propose a graph-based similarity calculation method in location-based social networks which improves the rating prediction performance of Singular Value Decomposition based collaborative filtering systems. We also propose a new rating prediction algorithm which increases the efficiency of rating prediction significantly. The methods are tested on check-in data of several users and the results are presented.

Keywords: Recommender Systems, Collaborative Filtering, Singular Value Decomposition, Feature Combination, Semantic Similarity, Graph Based Similarity, Similarity Recommendation

ÖZ

ÇOK-BOYUTLU YER BİLDİRİMİ VERİSİNDE BENZERLİK KULLANARAK VERİMLİ BİÇİMDE DEĞERLENDİRME TAHMİNİ

Uçar, Behlül
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Pınar Karagöz
Ortak Tez Yöneticisi : Prof. Dr. İ. Hakkı Toroslu

Eylül 2014, 64 sayfa

Son yıllarda konum tabanlı sosyal ağların kullanımı onlar üzerinde oluşan veri ile birlikte patlama yapmıştır. Bu veri tahminler yapmak üzere işlenmek için uygundur. Bu işleme için gerekli olan şeylerden birisi kullanılacak yöntemin çok büyük veri setlerine uygun olmasıdır.

Bu çalışmada Tekil Değer Ayrışımı bazlı Eşgüdümlü Filtreleme sistemlerinin performansını arttıran ve konum tabanlı sosyal ağlar üzerinde çalışan çizge tabanlı bir benzerlik hesaplama yöntemi öneriyoruz. Ayrıca değerlendirme tahmini işleminin verimliliğini belirgin biçimde arttıran yeni bir tahmin algoritması öneriyoruz. Metotlar birçok kullanıcının yer bildirim verisi üzerinde test edilmiş ve sonuçlar çalışmada sunulmuştur.

Anahtar Kelimeler: Önerici Sistemler, Ortaklaşa Filtreleme, Tekil Değer Ayrışımı, Özellik Harmanlama, Anlamsal Benzerlik, Graph Tabanlı Benzerlik, Benzerlik Önerileri

To my parents

Who always supported me in my thesis work

ACKNOWLEDGMENTS

The author wishes to express his deepest gratitude to his supervisor Assoc. Prof. Dr Pınar Karagöz and cosupervisor Prof. Dr. İ. Hakkı Toroslu for their guidance, advice, criticism, encouragements and insight throughout the research.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xiv

LIST OF FIGURES xv

LIST OF ALGORITHMS xix

LIST OF ABBREVIATIONS xx

CHAPTERS

1 INTRODUCTION 1

1.1 Contribution of the Thesis 3

1.2 Organization of the Thesis 3

2 BACKGROUND AND RELATED WORK 5

2.1 Collaborative Filtering 5

2.2 Content Based 7

2.3 Knowledge-Based 7

2.4 Hybrid Recommenders 7

2.5 Problems in Recommender Systems 9

2.6 Multi-Dimensional Recommendation 10

2.7 Graph-Based Methods 10

3 IMPROVED SVD-BASED RECOMMENDATION 13

3.1 Data Set Characteristics 13

3.2 Rating Estimation 14

3.3 Singular Value Decomposition Overview 14

3.4 SVD Prediction 15

3.4.1 Graph-Based Similarity Calculation 18

3.4.2 Weighted Average Based Rating Estimation . . 21

3.5 Running Example 21

3.5.1 Similarity Calculation 22

3.5.2 Collaborative Filtering Method 26

3.5.3 Content Based Method 28

3.5.4 Hybrid Method 31

4 SIMILARITY ESTIMATION BASED RECOMMENDATION . 35

4.1 Running Example 37

5 EVALUATION 39

5.1 Data Sets 39

5.2 Evaluation Metrics 40

5.3 Analysis for Parameter Optimization 40

5.4 Experimental Evaluation of the Improved SVD-Based Recommendation Method 42

5.4.1 Graph-Based Similarity Calculation 43

5.4.2 Weighted Average Calculation 46

5.4.3 Overall Comparison 50

5.5 Experimental Evaluation of the Similarity-Based Recommendation Method 53

6 CONCLUSION AND FINAL REMARKS 59

REFERENCES 61

LIST OF TABLES

TABLES

Table 2.1 Hybridization techniques 8

Table 3.1 Set of check-ins used in the example 22

Table 5.1 Rate of SimPred Results Against Hybrid SVD-Based Method 57

LIST OF FIGURES

FIGURES

Figure 3.1 Sample tensor shown as folds for each user 22

Figure 3.2 Calculated graph based similarities 25

Figure 3.3 Constructed matrix of CF method 26

Figure 3.4 Reduced-rank U and V matrices of CF method 27

Figure 3.5 Sorted similarity matrices of CF 27

Figure 3.6 Sorted similarity indices matrices of CF 27

Figure 3.7 Constructed matrix of CB method 29

Figure 3.8 Reduced-rank U and V matrices of CB method 29

Figure 3.9 Sorted similarity matrices of CB 30

Figure 3.10 Sorted similarity indices matrices of CB method 30

Figure 3.11 Constructed matrix of Hybrid method 31

Figure 3.12 Reduced-rank U and V matrices of Hybrid method 32

Figure 3.13 Sorted similarity matrices of Hybrid 33

Figure 3.14 Sorted similarity indices matrices of Hybrid 33

Figure 5.1 Impact of Neighborhood Size on MAE in Small Gowalla Data Set 41

Figure 5.2 Impact of Neighborhood Size on MAE in Medium Gowalla Data Set 41

Figure 5.3 Impact of Neighborhood Size on RMSE in Small Gowalla Data Set 42

Figure 5.4 Impact of Neighborhood Size on RMSE in Medium Gowalla Data Set 42

Figure 5.5 MAE of Similarity Calculation Methods in Gowalla Data Set 43

Figure 5.6 RMSE of Similarity Calculation Methods in Gowalla Data Set 44

Figure 5.7 MAE of Similarity Calculation Methods in Brightkite Data Set 44

Figure 5.8 RMSE of Similarity Calculation Methods in Brightkite Data Set 45

Figure 5.9 Preparation Times of SVD-based methods using different Similarity Matrices in Gowalla Data Set 45

Figure 5.10 Preparation Times of SVD-based methods using different Similarity Matrices in Brightkite Data Set 46

Figure 5.11 MAE of Average Calculation Methods in Small Gowalla Data Set 47

Figure 5.12 MAE of Average Calculation Methods in Small Brightkite Data Set 47

Figure 5.13 MAE of Average Calculation Methods in Medium Gowalla Data Set 48

Figure 5.14 MAE of Average Calculation Methods in Medium Brightkite Data Set 48

Figure 5.15 RMSE of Average Calculation Methods in Small Gowalla Data Set 49

Figure 5.16 RMSE of Average Calculation Methods in Small Brightkite Data Set 49

Figure 5.17 RMSE of Average Calculation Methods in Medium Gowalla Data Set 50

Figure 5.18 RMSE of Average Calculation Methods in Medium Brightkite Data Set 50

Figure 5.19 Overall Comparison of MAE Results in Small Gowalla Data Set 51

Figure 5.20 Overall Comparison of MAE Results in Small Brightkite Data Set 51

Figure 5.21 Overall Comparison of RMSE Results in Small Gowalla Data Set 52

Figure 5.22 Overall Comparison of RMSE Results in Small Brightkite Data Set 52

Figure 5.23 Overall Comparison of MAE Results in Medium Gowalla Data Set 53

Figure 5.24 Overall Comparison of RMSE Results in Medium Gowalla Data Set 53

Figure 5.25 MAE Comparison of SimPred method in Medium Gowalla Data Set 54

Figure 5.26 MAE Comparison of SimPred method in Medium Brightkite Data Set 54

Figure 5.27 RMSE comparison of SimPred method in Medium Gowalla Data Set 55

Figure 5.28 RMSE comparison of SimPred method in Medium Brightkite Data Set 55

Figure 5.29 Preparation Time Comparison of Proposed Methods in Medium Gowalla Data Set 56

Figure 5.30 Preparation Time Comparison of Proposed Methods in Medium Brightkite Data Set 56

LIST OF ALGORITHMS

ALGORITHMS

Algorithm 1 Graph-Based Similarity Calculation 20

Algorithm 2 SimPred Algorithm 36

LIST OF ABBREVIATIONS

CB	Content Based
CF	Collaborative Filtering
IR	Information Retrieval
MAE	Mean Absolute Error
PCA	Principal Component Analysis
RMSE	Root Mean Square Error
SimPred	Similarity Prediction
SVD	Singular Value Decomposition

CHAPTER 1

INTRODUCTION

In the early days of the internet, the amount of data present was limited. We have seen some stages of it which changed not only the amount of data contained, but also the methods of accessing them. First, we have seen the emerge of search engines such as Altavista or Google. They served as handy tools for exploring the data based on a text of interest which is entered as input to the system. These systems are called information retrieval (IR) systems in general. These work by matching the content of the documents to the query input and ranking the results according to some other criteria.

A possible improvement to the plain IR systems is personalizing the search results according to user. This is in a sense recommending new search results to the user. This recommendation is not limited to the plain search engines. The idea can be extended to electronic commerce sites, social networks, check-in sites, etc. In fact it can be extended to any site with a choice from the user side. Since every user's preference is different, from the overwhelming amount of information, the results can be personalized in order to fit better for their needs. This process is called recommendation and the systems doing it are called Recommender Systems (RS). RS'es are differentiated from IR systems by the criteria of "personalized" and "interesting and useful" [7].

More formally, recommender systems can be described as systems which try to make item suggestions to the users with the data of both that user and the other users. There may be explicit ratings given to the items or not, there may be friendships among users or not, these details depend on the nature of the application on which the recommender system is working. E-commerce applications as well as social media sites take advantage of recommender systems for different purposes. While an e-commerce site may use them for enabling the user to discover and buy more products; a social network application may employ them for just the discovery purpose and hence improving the engagement of the users to the application.

Since recommender systems can be employed by different applications, the definition of “item“ also changes according to the application. It can be a movie, a location, a product, say a book for example, music, a social activity or even people to add as friends.

With the advent of the smart phones and social networks installed in them as applications, we have seen an enormous increase in the amount of check-in data supplied by the users. This data contain not only the user and location dimensions, but also other dimensions such as the activity related to check-ins, time etc. We can utilize these dimensions by including them directly in our data structures as well as some relations within the dimensions such as similarity of items within those dimensions.

Recommending activities can be useful for not only the tourists or visitors who are new in a new city and do not know which activity and places are good for them, but also to the usual citizens of the city who want to discover new places from among many places in the city.

In [40], Zheng et al developed a collaborative algorithm using combined matrix factorization of matrices of different dimensions. Sattari et al showed in [31] that by merging matrices of different dimensions and applying Singular Value Decomposition on the merged matrix, they can outperform the method in [40]. In [30], the idea in [31] is developed further by using both intra-dimensional and inter-dimensional information.

1.1 Contribution of the Thesis

In this paper we aim to improve the idea in [30] by using a more robust similarity calculation method as well as a new rating estimation calculation formula. We also propose a new estimation method which does not use SVD and has a lower time complexity. Our contributions can be summarized as follows:

- We propose a graph-based similarity calculation method which is more robust and does not produce redundant entries in similarity matrices.
- We propose a weighted average calculation method instead of deviation based average method which improves the prediction performance.
- We propose a similarity based recommendation method which greatly reduces the run-time cost with an acceptable accuracy trade-off. In addition, since this method does not need any dimensionality reduction, it also avoids loss of information.

1.2 Organization of the Thesis

Chapter 2 gives some background information with related works. We compare different recommendation techniques that are proposed in the literature. We will present their advantages, disadvantages and the difference of our work.

In Chapter 3, Singular Value Decomposition (SVD) based recommendation system is described and our new similarity calculation technique is presented. A small example will be given as well.

In Chapter 4, our similarity based recommendation system will be presented and a small running example will be given again.

In Chapter 5 we present the evaluation results gathered and comment on them.

Finally in the last section we give a conclusion.

CHAPTER 2

BACKGROUND AND RELATED WORK

As the usage of social networks dramatically increased, there has emerged a huge data from these social networks which can be used for learning user preferences and making some recommendations to users [15]. These recommendations can also be used in e-commerce as discussed in [21].

There are different approaches and methods for making recommendations. Usually the common intuition in most of them is finding similar users and exploiting this similarity for prediction. We will analyze these methods according to their basic approach, data used, and method. We will compare their advantages and disadvantages, as well as the shortcomings and bottlenecks in them.

2.1 Collaborative Filtering

The systems which generate recommendations by using only the rating information are defined as collaborative systems. These systems find peers with similar history to the current user and generate recommendation from them. This selection of similar users is regarded as filtering and interaction of users as collaboration. Hence this method is called as Collaborative Filtering (CF) [19].

The method of collecting user preferences, i.e. rating information, depends on the nature of the applications. They can be collected explicitly or implicitly. In explicit data collection, users may be asked to rate the items on a scale, their favorite items may be used, or explicit likes of users such as in Facebook may be used. In implicit data collection, [25] shows that number of views for items can be used as rating. Similarly, in a location-based network, number of check-ins may be used as a liking indicator.

Collaborative filtering methods usually suffer from cold start, scalability and sparsity problems [20].

In a traditional CF system, there is no offline computation step. This brings the burden that its computation time scales with the number of users or items in the system [21]. A scalable RS should do the expensive calculation step offline and the online phase should not be dependent on the number of total items or users.

In order to overcome this scalability problem, different algorithms are tried in the literature. Matrix factorization methods are used to reduce the dimensionality of the matrices offline. For example, Singular Value Decomposition is used for this purpose. But these methods first still have a high complexity even for the offline phase, and secondly they cause losing some information in data. There are also some methods such as [40] which try to do this factorization with gradient descent algorithms. Since, gradient descent works by finding the local minima [37], they are not guaranteed to find the global minima.

There are also probabilistic models which utilizes probability theory to create relationships between users and items to make recommendations. Since they calculate the probabilities offline, they can be scalable in the online phase.

For example, [23] uses Bayesian classification for binary rating estimation. They make the assumption that every user's votes are independent from each other. Then they try to calculate the probabilities of a user's negative or positive vote. Note that their method is for 2 dimensional data and discrete preferences.

As a summary, it is essential to use some data-reduction technique to overcome the scalability problem in CF and in general in all recommender systems.

2.2 Content Based

As described in the previous section, CF systems use only the rating information gathered from users. But in Content Based (CB) systems, we also employ the profile information for users and other dimensions. The system tries to recommend items that match the user profile [19]. For example, [12] shows building user profiles in different ways using the keywords from websites that the users have visited in the past. [26] argues some other techniques which work by classifying users such as decision trees and machine learning to create user profiles.

2.3 Knowledge-Based

[8] defines knowledge-based recommenders as recommenders which suggest products based on a user's needs and preferences. They respond to the user's immediate needs and they do not need retraining when preferences change. Training is done on the knowledge-base instead. The drawback of them is that knowledge base should be constructed very precisely such that users should be able to select their preferences from them.

2.4 Hybrid Recommenders

In a broad sense, hybrid recommenders are defined as the combination of two different recommender systems to get more accurate results. In this way, in an area where one RS has shortcomings, the other RS can come to help. It is also possible that this combination can give worse results if the two algorithms are not filling each other's empty blocks.

Note that the hybridized systems do not necessarily have to employ different techniques but they may also. For example, two CB recommender systems could work together as well as one CF and CB recommender systems. [12] lists seven different hybridization techniques, they are shown in Table 2.1.

Table2.1: Hybridization techniques

Mixture method	**Description**
Weighted	The score of different recommendation components are combined numerically.
Switching	The system chooses among recommendation components and applies the selected one.
Mixed	Recommendations from different recommenders are presented together.
Feature Combination	Features derived from different knowledge sources are combined together and given to a single recommendation algorithm.
Feature Augmentation	One recommendation technique is used to compute a feature or set of features, which is then part of the input to the next technique.
Cascade	Recommenders are given strict priority, with the lower priority ones breaking ties in the scoring of the higher ones.
Meta-level	One recommendation technique is applied and produces some sort of model, which is then the input used by the next technique.

2.5 Problems in Recommender Systems

In this section we will look into some of the general problems of recommender systems.

Cold start is defined as the need for having a large amount of data on a user to make accurate recommendations. This is normal since CF systems do not take user profile into account. This problem is also valid for other methods such as Content Based and Hybrid but since we take other information into account as well, it is lighter. Still, for many recommender systems it is not very accurate to to make recommendation before knowing the user's behaviours [3].

Scalability problem is defined as the memory and run-time requirement for running the recommendation algorithms. This is a critical problem in systems with millions, sometimes even billions, of users [32, 29]. Scalability is important in both the online and offline phases of the recommender systems. That is preprocessing of the data set should be computationally reasonable to complete in industrially acceptable time and efficiency of the online phase which will be active on user query time should be quick enough not to keep users waiting for the recommendation results. Some methods in the literature uses dimensionality reduction methods such as PCA or SVD. For example [13] proposes a recommender for jokes using PCA and a clustering. Graph-theoretic approaches to recommender systems are also common such as in [4].

Sparsity problem can be put as having a very low feedback from users. Also, for the user whose tastes are unusual compared to the rest of the population there will not be any other users who are particularly similar, leading to poor recommendations [6]. This problem is usually encountered in systems that use explicit data collection and can be alleviated with implicit collection. User profile information may also be used to make demographic filtering such as in [27]. Some works [16] uses similarity between users in the recommendation phase to alleviate sparsity problem.

There is also a flexibility problem in recommender systems. Most recommender systems recommend general items, that is they do not support filtered recommendations. [2] identifies this problem and proposes a Recommendation Query Language (RQL) for querying recommendations. However the underlying recommendation engine should be able to support aggregated recommendations to make RQL work. OLAP-based approach [9] is proposed to support aggregated recommendation in multi-dimensional domains [1]. This is an ongoing research area in recommender systems.

There are other problems and extensions such as using multi-criteria ratings and non-intrusiveness for the user. These extensions all constitute a different research area in recommender systems, we will not go into detail of them.

2.6 Multi-Dimensional Recommendation

If we come to the multi-dimensional domain, some of these methods use 2-D matrices whereas some see the data as a multi-dimensional tensor. [40] is one example which uses different 2-D matrices for different dimensional information and tries to predict the unknown values by Collective Matrix Factorization. It was mentioned earlier that it uses gradient descent. In [39], they follow a tensor based approach in which they construct a tensor and define an objective function from that tensor. Then they use gradient descent to solve the problem as in [40]

In [31] and [30], they first construct a merged matrix with different dimensions and employ Singular Value Decomposition on the merged matrix. At the end they get two reduced rank matrices. By using these reduced-rank matrices, they find the most similar items for the item of interest and estimate the rating from similar items.

2.7 Graph-Based Methods

Structures of social networks are suitable for considering them as graph, $G = (V, E)$ where V is the set of vertices and E is the set of edges. Items can be considered as nodes and relations between them as edges. By using this structure, algorithms like Random Walk with Restart (RWR) can be applied on these graphs. For example, [35] uses RWR algorithm with some additional tagging information on a social network graph. They work by creating a bipartite graph and travelling on it. [17] creates a user-book graph and assigns similarities between edges to recommend books to users. Note that these methods work only in 2-D domains.

One of the most important things in graph-based methods is calculating the similarities between items. There are different approaches in the literature for finding similarity. For example, [38] uses a clustering algorithm named DBSCAN to cluster items and find similarity between them using cosine similarity. [41] creates sequences from the data and checks the number of common elements in them. These approaches suffer from the time complexity in very large data sets. In this work, we will follow a graph-based approach which is also suitable for large data sets.

In this thesis in the first part we present a generic graph-based similarity calculation scheme which can be thought as probabilities in a probabilistic system. The proposed method is robust and suitable for very big data sets. In addition a new formula for the method in [30] is tried. In the second part we present a new similarity based algorithm which does not use SVD at all.

CHAPTER 3

IMPROVED SVD-BASED RECOMMENDATION

In this chapter we describe the base method which is also the subject of [30]. We developed a new similarity extraction method which can be used in this method as well as the Similarity Recommendation method which will be presented in the next chapter.

We will first describe the data set and the characteristics of it. Since we worked with data with no explicit user ratings, our method for rating estimation for visited places will be presented next. Then we will give information about Singular Value Decomposition and use of it in recommendation. After that we will describe our graph based similarity calculation technique and in the last section a small running example will be given.

3.1 Data Set Characteristics

In this section we will describe the general characteristics of the data sets used by proposed methods. We believe describing them before explaining the algorithm will be helpful for easier understanding.

In location based social networks, primary means of information is the check-ins of users. Check-ins usually have basic information such as date and time and location and may have additional information such as the activity performed. Our methods use these information as dimensions. Although the proposed methods are generic in terms of dimensions, specifically we use user, location and activity information.

3.2 Rating Estimation

Since we did not have explicitly given ratings from users, we had to estimate them from the past activities of the users. For that purpose, we used the check-in frequencies of the users for each item of each dimensions in the data set. For our case, that became ratings of users for doing an activity at a location.

Since we gave rating according to past activity, only the items which user experienced had ratings. The others are unknown and the subject of the rating prediction algorithm.

For each point in the multi-dimensional space, we checked the number of check-ins and got a rating according to Equation 3.1.

$$R(i) = 1 + \ln f_c \tag{3.1}$$

In this equation i is an item in the tensor and f_c is the check-in frequency for that item.

3.3 Singular Value Decomposition Overview

Using Singular Value Decomposition, we can factorize an mxn matrix into three matrices in the following way:

$$R_{mxn} = U_{mxr} S_{rxr} V_{rxn}^T \tag{3.2}$$

Here, U is the left singular vector and V is the right singular vector. They are orthogonal to each other. In this equation, r is the rank of the matrix R. Columns of U and V come from the eigenvectors of RR^T and $R^T R$, respectively [28].

S is a rectangular diagonal matrix with non-negative real numbers being on the diagonal. Its diagonal entries are called singular values of R. All diagonal entries are positive and stored in decreasing order of magnitude. According to Eckart-Young theorem, we can get a low-rank approximation of R using its SVD [11].

We can utilize SVD in recommender systems in two different aspects. First, it helps us capture the latent relationships between the data. Second, it's a method to reduce the dimensionality of data, which is essential for scalability [28].

Capturing the latent relationships is done by comparing the rows U or V and extracting a similarity between each other of them. Note that in order to this efficiently a dimensionality reduction is needed again.

The dimensionality reduction is done by selecting the k largest values of S and k columns of U and V. By multiplying these trimmed matrices we can get a low-rank approximation of R in the following way:

$$R_k = U_{mxk} S_{kxk} V^T_{kxn} \tag{3.3}$$

Here, R_k is the low-rank approximation of R with rank k. Note that in this approach we lose some data. We determine the loss percentage of data by the sum of singular values in S. Since singular values are sorted in descending order, we divide the sum of values in trimmed matrix by sum of values of whole matrix. This is shown in the following equation:

$$p = \frac{\sum_{i=1}^{k} S_k}{\sum_{all} S_k} \tag{3.4}$$

3.4 SVD Prediction

As shown in [30], we can reduce the multi-dimensional tensor into a 2-D matrix in 3 different ways; Collaborative Filtering, Content Based and Hybrid. This is shown in Equations 3.5, 3.6 and 3.7.

$$M_{CF(a+l+u)x(a+l+u)} = \begin{bmatrix} 0_{axa} & 0_{axl} & 0_{axu} \\ LA & 0_{lxl} & 0_{lxu} \\ UA & UL & 0_{uxu} \end{bmatrix} \tag{3.5}$$

$$M_{CB(a+l+u)x(a+l+u)} = \begin{bmatrix} AA & 0_{axl} & 0_{axu} \\ 0_{lxa} & LL & 0_{lxu} \\ 0_{uxa} & 0_{ul} & UU \end{bmatrix} \tag{3.6}$$

$$M_{Hyb(a+l+u)x(a+l+u)} = \begin{bmatrix} AA & 0_{axl} & 0_{axu} \\ LA & LL & 0_{lxu} \\ UA & UL & UU \end{bmatrix} \tag{3.7}$$

Here, AA, UU and LL are activity-activity, user-user and location-location similarity matrices, respectively. In the next section we will present how they are calculated. LA, UA and UL matrices are location-activity, user-activity and user-location matrices. They are the average values gathered from the original tensor. For example $LA[0, 0]$ is the average rating for location l_0 and activity a_0 gathered over all the users. Note that M_{CF} has only the information obtained from the tensor whereas M_{CB} has only similarity matrices, and M_{Hyb} has all of them, being the hybrid method.

We then apply SVD to reduce the rank of M and reveal the latent semantic indexing of the data. Since SVD is a lossy method, we have to select a percentage of the data to keep in the matrix. After applying SVD, we decompose M into U, S, and V matrices satisfying $M_k = U_{rxk} S_{kxk} V^T_{kxs}$ where M_k is the k-rank reduced representation of M [11].

After applying SVD, we can divide U row-wise and V column-wise to get different matrices for Activity, Location and User. So we can treat U and V as follows:

$$U_{kxr} = \begin{bmatrix} U_{rAct} \\ U_{rLoc} \\ U_{rUser} \end{bmatrix} \tag{3.8}$$

$$V_{rxk} = \begin{bmatrix} V_{rAct} & V_{rLoc} & V_{rUser} \end{bmatrix} \tag{3.9}$$

Then using these matrices we got from dividing, we can calculate similarity matrices for the dimensions. For that purpose, cosine similarity between rows for U_{rAct}, U_{rLoc}, U_{rUser} or columns for V_{rAct}, V_{rLoc}, V_{rUser} is used. We get 6 similarity matrices in total at the end.

After we obtain similarity matrices, we can use them to predict missing entries in the original tensor. For each dimension of the tensor, we freeze the indices of other dimensions and find similar objects using the similarity matrices we got. In our case, for an entry (u_i, l_j, a_k) we have 6 possible values as follows:

$$P^U_{User} = r_{u_i} + \frac{\sum_{p=1}^{m}(T(u_p, l_j, a_k) - r_{u_p}) * sim_{U_{rUser}}(u_i, u_p)}{\sum_{p=1}^{m} sim_{U_{rUser}}(u_i, u_p)} \tag{3.10}$$

$$P^V_{User} = r_{u_i} + \frac{\sum_{p=1}^{m}(T(u_p, l_j, a_k) - r_{u_p}) * sim_{V_{rUser}}(u_i, u_p)}{\sum_{p=1}^{m} sim_{V_{rUser}}(u_i, u_p)} \tag{3.11}$$

$$P^U_{Loc} = r_{l_j} + \frac{\sum_{p=1}^{m}(T(u_i, l_p, a_k) - r_{l_p}) * sim_{U_{rLoc}}(l_i, l_p)}{\sum_{p=1}^{m} sim_{U_{rLoc}}(l_i, l_p)} \tag{3.12}$$

$$P^V_{Loc} = r_{l_j} + \frac{\sum_{p=1}^{m}(T(u_i, l_p, a_k) - r_{l_p}) * sim_{U_{rAct}}(l_i, l_p)}{\sum_{p=1}^{m} sim_{V_{rLoc}}(l_i, l_p)} \tag{3.13}$$

$$P^U_{Act} = r_{a_k} + \frac{\sum_{p=1}^{m}(T(u_i, l_j, a_p) - r_{a_p}) * sim_{U_{rAct}}(a_i, a_p)}{\sum_{p=1}^{m} sim_{U_{rAct}}(a_i, a_p)} \tag{3.14}$$

$$P^V_{Act} = r_{a_k} + \frac{\sum_{p=1}^{m}(T(u_i, l_j, a_p) - r_{a_p}) * sim_{V_{rAct}}(a_i, a_p)}{\sum_{p=1}^{m} sim_{V_{rAct}}(a_i, a_p)} \tag{3.15}$$

m is the set of neighbors in these equations. For each value, we select the most similar items from the similarity matrices and put them in m. Also,we exclude from m the items that have 0 value in the tensor. For example, for calculating P^U_{User}, we select the most similar rows from U_{rUser} such that $T[u_p, l_j, a_k] \neq 0$.

As stated earlier, we can construct the merged matrix in three different ways. The structure of the merged matrix also affects the values we can use in prediction. This is because since some parts of the merged matrix become zero in different methods, the corresponding parts in the U_r and V_r matrices also become zero matrices. For Collaborative filtering, Content Based and Hybrid methods, we use the following prediction formulae:

$$\hat{T}_{CF}(u_i, l_j, a_k) = \frac{P^U_{User} + P^U_{Loc} + P^V_{Loc} + P^V_{Act}}{4} \tag{3.16}$$

$$\hat{T}_{CB}(u_i, l_j, a_k) = \frac{P^U_{User} + P^U_{Loc} + P^U_{Act}}{3} \tag{3.17}$$

$$\hat{T}_{Hyb}(u_i, l_j, a_k) = \frac{P^U_{User} + P^U_{Loc} + P^U_{Act} + P^V_{User} + P^V_{Loc} + P^V_{Act}}{6} \tag{3.18}$$

3.4.1 Graph-Based Similarity Calculation

So far we have described the method in [30] and we have skipped the similarity calculation phase. In this section we will present the new similarity calculation methods and a new formula for rating predictions in the next section. These are our contributions to the original algorithm.

In [30], the similarity calculation methods used are simple. For location similarity, they simply use the distance between locations. This may be misleading in reality because there may be two very different locations even next to each other. For activity similarity, they treat the hour of the day as activity and difference between hours as the similarity between two items. Although this may make sense with respect to the fact that the hour that activities are performed will be near, this is not a real similarity measure either. For user similarity, they use number of common friends. This may be again misleading for example in the case of two students from the same university and hence having the same friends. We propose a new similarity calculation method which is general and constructed from check-in history.

In order to calculate the similarities inside a dimension d_1, we represent d_1 as a graph and use another dimension d_2 to calculate edge weights between nodes in this graph. We define a session array which is empty at the beginning. Session array keeps the previous check-ins which are not older than a predefined interval. Then we sort the check-ins with respect to d_2 primarily and date secondarily. Then for each check-in, we compare each check-in in the session array with that check-in. If the check-in in the session array is older than the check-in in hand over a threshold, we remove it from the session. Otherwise we increase the edge weight between d_1 values of these check-ins. After we compared all the check-ins, we add the check-in in the hand to the session array and proceed to the next check-in. If we encounter a different d_2 item, we reset the session array. After we finish processing all the check-ins, we normalize the weights of edges such that sum of a node's edges makes 1. In this way we calculate the similarity inside d_1. The algorithm is shown as pseudo code in Algorithm 1.

In our case, we calculated user-user similarity using location dimension as d_2, location-location and activity-activity similarities using user dimension as d_2. For user-user similarity, we iterated over the check-ins done in locations, for the users of check-ins done in a neighborhood of 30 days, we increased the edge weight of them by 1. In this way we tried to find "users with same tastes".

Time complexity of this approach is $O(nlog(n))$ for sorting and $O(nm)$ for finding similarities, where n is the number of check-ins and m is the number items in the session. Number of items in a session is much less than number of total check-ins, in other words $m \ll n$. Hence, the general complexity of graph-based similarity calculation is $O(nlog(n))$.

One other point worth noting about the new calculation method is that it does not produce a dense similarity matrix. The location-location and user-user similarities calculated in [30] produces dense matrices in the sense that they assign similarity between all items. This not only increases the run-time complexity of similarity calculation to $O(n^2)$, but also increases the run-time cost of SVD-based methods, since working with denser matrices requires more work.

As described in the previous section, these similarity matrices are used in Content Based and Hybrid calculation methods. For Collaborative Filtering, we only use matrices obtained from the tensor.

```
**Data**: checkins : set of checkins
**Result**: itemGraph : a sparse similarity matrix
session = []
init itemGraph as a sparse matrix
itemDates = {} prevD2Item = None prevDate = None
sort checkins according to $d_2$ and date
for each ci in checkins do
    itemDates[ci.d1] = ci.time
    if prevD2Item != ci.d2Item then
        sesUsers = []
    end
    start = 0
    while session is not empty and ci is not older than session[0] do
        erase first element in session
    end
    if ci.item in session then
        start = session.index(ci.item)
        session.remove(ci.item)
    end
    for i in range(start, len(session) ) do
        item = session[i] if ci.item not in itemGraph then
            itemGraph[ci.item] = {}
        end
        if item not in itemGraph then
            itemGraph[item] = {}
        end
        val = 1
        itemGraph[ci.item][item] = itemGraph[ci.item][item]+val
            if item in itemGraph[ci.item] else val
        itemGraph[u][ci.item] = itemGraph[item][ci.item]+val
            if ci.item in itemGraph[item] else val
    end
    session.append(ci.item)
    prevDate = ci.t
    prevD2Item = ci.d2Item
end
normalize edge weights of itemGraph
```

Algorithm 1: Graph-Based Similarity Calculation

3.4.2 Weighted Average Based Rating Estimation

Our second modification to the original method is a new rating estimation formula. Original method takes the average standard deviations of item ratings into account. In other words, it adds the weighted average of deviations gathered from the neighbors to the average ratings for that item. Although [14] states that average of standard deviations perform better, in our case taking weighted averages of the ratings of the neighbors performed better. This is maybe because of the fact that in deviation average, average deviation of the neighbor items are added to the average of the item of prediction. These may not be correlated at all. For example, the item of prediction may have ratings close to 5 whereas the neighbors close to 3. Adding the deviation to the average may not make sense in that case. One may argue that direct weighted average method does not solve this problem either. However, it smooths this effect and empirical results support this argument as well. The new formulae for the values are as follows:

$$P^U_{User} = \frac{\sum_{p=1}^{m} T(u_p, l_j, a_k) * sim_{U_{rUser}}(u_i, u_p)}{\sum_{p=1}^{m} sim_{U_{rUser}}(u_i, u_p)} \tag{3.19}$$

$$P^V_{User} = \frac{\sum_{p=1}^{m} T(u_p, l_j, a_k) * sim_{V_{rUser}}(u_i, u_p)}{\sum_{p=1}^{m} sim_{V_{rUser}}(u_i, u_p)} \tag{3.20}$$

$$P^U_{Loc} = \frac{\sum_{p=1}^{m} T(u_i, l_p, a_k) * sim_{U_{rLoc}}(l_i, l_p)}{\sum_{p=1}^{m} sim_{U_{rLoc}}(l_i, l_p)} \tag{3.21}$$

$$P^V_{Loc} = \frac{\sum_{p=1}^{m} T(u_i, l_p, a_k) * sim_{U_{rAct}}(l_i, l_p)}{\sum_{p=1}^{m} sim_{V_{rLoc}}(l_i, l_p)} \tag{3.22}$$

$$P^U_{Act} = \frac{\sum_{p=1}^{m} T(u_i, l_j, a_p) * sim_{U_{rAct}}(a_i, a_p)}{\sum_{p=1}^{m} sim_{U_{rAct}}(a_i, a_p)} \tag{3.23}$$

$$P^V_{Act} = \frac{\sum_{p=1}^{m} T(u_i, l_j, a_p) * sim_{V_{rAct}}(a_i, a_p)}{\sum_{p=1}^{m} sim_{V_{rAct}}(a_i, a_p)} \tag{3.24}$$

3.5 Running Example

In this section, we present a running example for the proposed SVD-based methods. In order to show the steps of the algorithms, we will use a tensor with 5 users, 3 locations and 3 activities. This tensor can be seen in Figure 3.1. We show the tensor as a matrix for each user, since it is a 3 dimensional structure.

1	0	0	0
2.38629	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

0	0	0	0
2.79176	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

1	0	0	0
2.38629	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

0	0	0	0
0	0	0	0
0	3.07944	0	0
0	0	2.79176	0
0	0	0	2.94591

2.60944	0	0	0
1.69315	0	0	0
0	3.07944	0	0
0	0	1	0
0	0	0	1

Figure 3.1: Sample tensor shown as folds for each user

3.5.1 Similarity Calculation

Similarity values are used in CB and Hybrid methods. To illustrate how we calculate the similarity values, we first present the check-ins we use as input in Table 3.1. This list is taken from the actual data set as a very small sample of it.

Table3.1: Set of check-ins used in the example

User	**Loc**	**Time**	**Activity**
1	640452	2010-10-02T18:58:55	Doctor's Office
1	102499	2010-08-29T01:40:49	Seafood Restaurant
1	260236	2010-05-06T19:52:33	Cocktail Bar
1	260236	2010-05-16T19:52:33	Cocktail Bar
1	260236	2010-06-26T19:52:33	Cocktail Bar
1	260236	2010-08-16T20:21:53	Cocktail Bar
1	192704	2010-10-20T06:22:17	Cocktail Bar
1	33509	2010-10-18T20:38:01	Coffee Shop
2	640452	2010-10-03T18:58:55	Doctor's Office
2	102499	2010-08-30T01:40:49	Seafood Restaurant

Table 3.1 – Continued from previous page

User	Loc	Time	Activity
2	260236	2010-05-06T19:52:33	Cocktail Bar
2	260236	2010-05-16T19:52:33	Cocktail Bar
2	260236	2010-06-27T19:52:33	Cocktail Bar
2	260236	2010-08-16T20:21:53	Cocktail Bar
2	192704	2010-11-20T06:22:17	Cocktail Bar
2	33509	2010-12-18T20:38:01	Coffee Shop
3	260236	2010-05-06T19:52:33	Cocktail Bar
3	260236	2010-05-16T19:52:33	Cocktail Bar
3	260236	2010-06-29T19:52:33	Cocktail Bar
3	260236	2010-06-30T19:52:33	Cocktail Bar
3	260236	2010-07-30T19:52:33	Cocktail Bar
3	260236	2010-09-10T19:52:33	Cocktail Bar
4	640452	2010-10-02T18:58:55	Doctor's Office
4	102499	2010-08-29T01:40:49	Seafood Restaurant
4	260236	2010-08-16T20:21:53	Cocktail Bar
4	260236	2010-06-26T19:52:33	Cocktail Bar
4	192704	2010-10-20T06:22:17	Cocktail Bar
4	192704	2010-11-20T06:22:17	Cocktail Bar
4	192704	2010-11-21T06:22:17	Cocktail Bar
4	192704	2010-11-23T06:22:17	Cocktail Bar
4	192704	2010-11-25T06:22:17	Cocktail Bar
4	33509	2010-01-18T20:38:01	Coffee Shop
4	33509	2010-02-18T20:38:01	Coffee Shop
4	33509	2010-03-15T20:38:01	Coffee Shop
4	33509	2010-04-18T20:38:01	Coffee Shop
4	33509	2010-05-18T20:38:01	Coffee Shop
4	33509	2010-06-18T20:38:01	Coffee Shop
4	33509	2010-07-18T20:38:01	Coffee Shop
4	33509	2010-08-18T20:38:01	Coffee Shop
5	640452	2010-10-02T18:58:55	Doctor's Office

Table 3.1 – Continued from previous page

User	**Loc**	**Time**	**Activity**
5	640452	2010-10-03T18:58:55	Doctor's Office
5	640452	2010-10-04T18:58:55	Doctor's Office
5	640452	2010-10-05T18:58:55	Doctor's Office
5	640452	2010-10-06T18:58:55	Doctor's Office
5	640452	2010-10-07T18:58:55	Doctor's Office
5	102499	2010-08-20T01:40:49	Seafood Restaurant
5	102499	2010-08-21T01:40:49	Seafood Restaurant
5	102499	2010-08-22T01:40:49	Seafood Restaurant
5	102499	2010-08-23T01:40:49	Seafood Restaurant
5	102499	2010-08-29T01:40:49	Seafood Restaurant
5	102499	2010-08-30T01:40:49	Seafood Restaurant
5	102499	2010-09-10T01:40:49	Seafood Restaurant
5	33509	2010-10-01T20:38:01	Coffee Shop
5	33509	2010-10-02T20:38:01	Coffee Shop
5	33509	2010-10-03T20:38:01	Coffee Shop
5	33509	2010-10-04T20:38:01	Coffee Shop
5	33509	2010-10-05T20:38:01	Coffee Shop
5	33509	2010-10-06T20:38:01	Coffee Shop
5	33509	2010-10-18T20:38:01	Coffee Shop
5	33509	2010-10-19T20:38:01	Coffee Shop

As an example calculation, we will try to find the similarity between users 1 and 2. Note that we will not follow the steps of the algorithm exactly but get the unnormalized similarity between user 1 and 2 only. This is for brevity of the example.

For getting the user-user similarity, we first sort the check-ins with respect to the locations primarily and dates secondarily. For location 640452, User 1 has a check-in on 2010-10-02. User 2 has a check-in on 2010-10-03 as well. Since the time between these check-ins are not greater than 30 days, we add 1 to the value of the edge between user 1 and 2. From the contribution of location 102499, we add 1 more to the edge value. We proceed to location 260236. We first get user 1's earliest check-in, then user 2's check-in on 2010-05-06 contributes 1. We encounter user 1's check-in on 2010-05-16, we increment the weight by 1 because of user 2's earlier check-in and remove the earlier check-in of user 1. Then we encounter user 2's check-in on 2010-05-16. We update for user 2 and increment by 1. Then we encounter user 1's check-in on 2010-06-26. Since there is no previous check-in in 30 days period, no increment is done. Then we encounter user 2's check-in on 2010-06-27 and increment the weight by 1. At the time of user 1's check-in on 2010-10-18, again there is no previous check-in. And user 2's check-in on 2010-08-16 increases the weight by one. The check-ins to location 192704 contributes 1 since check-in of user 2 is in the border of 30 days. But the check-ins to location 33509 does not contribute. At the end, we get an edge weight of 8 between user 1 and 2.

After we calculate weights for all edges of user 1, we normalize them and get the final similarity values. Note that this similarity calculation is not symmetric, meaning that similarity of user 1 to user 2 may be different from similarity of user 2 to user 1. The normalized final similarity values are given in Figure 3.2.

0	0.28	0.32	0.2	0.2
0.438	0	0.375	0	0.188
0.348	0.261	0	0.174	0.217
0.417	0	0.333	0	0.25
0.312	0.188	0.312	0.188	0

(a) User-User similarity

0	0	0.5	0.5	0
0	0	0.571	0	0.429
0.0952	0.19	0	0.619	0.0952
0.125	0	0.812	0	0.0625
0	0.5	0.333	0.167	0

(b) Location-Location similarity

0	0.4	0	0.6
0.667	0	0	0.333
0	0	0	0
0.75	0.25	0	0

(c) Activity-Activity similarity

Figure 3.2: Calculated graph based similarities

3.5.2 Collaborative Filtering Method

For CF method, we use just the data obtained from the tensor. We construct a 2-D matrix according to Equation 3.5. Then our constructed 2-D matrix becomes as in Figure 3.3. These are the average values gathered from tensor shown in 3.1. From the constructed matrix, we get reduced rank matrices which are shown in Figure 3.4. Using cosine distance metric, we then gather similarity matrices. Sorted similarity matrices with their indices are presented in Figures 3.5 and 3.6.

0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1.536	0	0	0	0	0	0	0	0	0	0	0	0	0
2.314	0	0	0	0	0	0	0	0	0	0	0	0	0
0	2.04	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1.448	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1.486	0	0	0	0	0	0	0	0	0	0
1.693	2.792	3.386	0	1	0	1	0	2.609	0	0	0	0	0
4.303	0	0	0	2.386	2.792	2.386	0	1.693	0	0	0	0	0
1.54	3.079	1	0	1	0	1	3.079	3.079	0	0	0	0	0
0	2.792	1	1	1	0	1	2.792	1	0	0	0	0	0
0	1	2.946	1	1	0	1	2.946	1	0	0	0	0	0

Figure 3.3: Constructed matrix of CF method

0.788	0.0298	0.488
0.192	0.815	0.391
0.296	0.147	0.546
0.391	0.317	0.382
0.0608	0.315	0.378

(a) U-user

0.0252	0.172	0.0557
0.0379	0.259	0.0839
0.0415	0.11	0.0955
0.296	0.0629	0.0546
0.0772	0.0274	0.0115

(b) U-loc

0.127	0.188	0.273
0.162	0.376	0.109
0.127	0.188	0.273
0.66	0.375	0.386
0.112	0.0356	0.438

(c) V-loc

0.0541	0.677	0.363
0.0672	0.325	0.468
0.675	0.262	0.377
0.172	0.111	0.0777

(d) V-act

Figure 3.4: Reduced-rank U and V matrices of CF method

0.572	0.474	0.149	0.14	0
0.572	0.338	0.173	0.0914	0
0.338	0.149	0.144	0.0803	0
0.173	0.144	0.14	0.0803	0
0.474	0.144	0.144	0.0914	0

(a) U-user-sim

0.621	0.517	0.0863	0	0
0.621	0.517	0.0863	0	0
0.476	0.416	0.0863	0.0863	0
0.621	0.621	0.476	0.00923	0
0.517	0.517	0.416	0.00923	0

(b) U-loc-sim

0.195	0.141	0.127	0	0
0.588	0.196	0.195	0.195	0
0.195	0.141	0.127	0	0
0.337	0.196	0.141	0.141	0
0.588	0.337	0.127	0.127	0

(c) V-loc-sim

0.442	0.33	0.11	0
0.345	0.33	0.11	0
0.442	0.345	0.0243	0
0.33	0.33	0.0243	0

(d) V-act-sim

Figure 3.5: Sorted similarity matrices of CF

1	4	2	3	0
0	2	3	4	1
1	0	4	3	2
1	4	0	2	3
0	3	2	1	4

(a) U-user-sim-ind

3	4	2	0	1
3	4	2	0	1
3	4	0	1	2
1	0	2	4	3
1	0	2	3	4

(b) U-loc-sim-ind

1	3	4	2	0
4	3	2	0	1
1	3	4	2	0
4	1	2	0	3
1	3	2	0	4

(c) V-user-sim-ind

2	3	1	0
2	3	0	1
0	1	3	2
1	0	2	3

(d) V-act-sim-ind

Figure 3.6: Sorted similarity indices matrices of CF

For showing the prediction process, we select a random entry from the tensor and try to predict its rating value. We select the entry with position $(1, 1, 0)$ in the tensor, whose value is 2.79176 and denote it with $e_{(1,1,0)}$.

We can get 4 different values using the 4 reduced-rank matrices we have. For each of them, we freeze the indices in the dimensions other than the dimension of the matrix. So for example when using U-user matrix, we freeze location and activity dimension indices. We then check for the most similar users using the corresponding row of the matrix and use the corresponding entry in the tensor for prediction. Note that for brevity, we will work with a neighborhood size of 1 throughout the examples.

For user 1, the most similar user is user 0. Its similarity is 0.575. It's tensor value $T_{0,1,0}$ is not zero and 2.38629. Then we can use it. The prediction becomes:

$$P^{u}_{User} = \frac{2.38629 * 0.575}{0.575} = 2.38629$$

$$P^{u}_{Loc} = undefined, P^{v}_{Loc} = undefined, P^{v}_{Act} = undefined$$

$$\hat{T}(1, 1, 0) = 2.38629$$

Note that since there is no other entry of this user in location and activity dimensions, P^{u}_{Loc}, P^{v}_{Loc} and P^{v}_{Act} were undefined. We use only P^{u}_{User} for prediction. At the end, we get the predicted value of 2.38629

3.5.3 Content Based Method

For Content-Based method, our 2-D constructed matrix becomes as in Equation 3.6. This can be seen in Figure 3.7. Corresponding reduced rank matrices, similarity matrices and sorted similarity matrices are shown in Figures 3.8 to 3.10.

0	0.4	0	0.6	0	0	0	0	0	0	0	0	0	0
0.67	0	0	0.33	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.75	0.25	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0.5	0.5	0	0	0	0	0	0
0	0	0	0	0	0	0.57	0	0.43	0	0	0	0	0
0	0	0	0	0.095	0.19	0	0.62	0.095	0	0	0	0	0
0	0	0	0	0.12	0	0.81	0	0.062	0	0	0	0	0
0	0	0	0	0	0.5	0.33	0.17	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0.28	0.32	0.2	0.2
0	0	0	0	0	0	0	0	0	0.44	0	0.38	0	0.19
0	0	0	0	0	0	0	0	0	0.35	0.26	0	0.17	0.22
0	0	0	0	0	0	0	0	0	0.42	0	0.33	0	0.25
0	0	0	0	0	0	0	0	0	0.31	0.19	0.31	0.19	0

Figure 3.7: Constructed matrix of CB method

0.34	$3.1e-13$	$1.8e-16$
0.53	$4.9e-13$	$2.4e-16$
0.38	$3.5e-13$	$2e-16$
0.52	$4.8e-13$	$1.8e-16$
0.44	$4.1e-13$	$2.3e-16$

(a) U-user

0	$5.6e-17$	0.5
$1.7e-16$	$5.6e-17$	0.48
$2.8e-17$	$5.6e-17$	0.22
$1.1e-16$	$1.1e-16$	0.59
$5.6e-17$	0	0.35

(b) U-loc

$2.9e-13$	0.31	$2e-16$
$6.1e-13$	0.66	$2.5e-16$
$6.2e-17$	$3.5e-17$	$7.3e-17$
$6.3e-13$	0.68	$1.5e-16$

(c) U-act

Figure 3.8: Reduced-rank U and V matrices of CB method

0.19	0.18	0.099	0.041	0
0.19	0.15	0.092	0.015	0
0.15	0.14	0.058	0.041	0
0.18	0.14	0.077	0.015	0
0.099	0.092	0.077	0.058	0

(a) U-user-sim

0.27	0.14	0.099	0.019	0
0.25	0.12	0.12	0.019	0
0.37	0.27	0.25	0.13	0
0.37	0.24	0.12	0.099	0
0.24	0.14	0.13	0.12	0

(b) U-loc-sim

0.37	0.35	0.31	0
0.66	0.35	0.014	0
0.68	0.66	0.31	0
0.68	0.37	0.014	0

(c) U-act-sim

Figure 3.9: Sorted similarity matrices of CB

1	3	4	2	0
0	2	4	3	1
1	3	4	0	2
0	2	4	1	3
0	1	3	2	4

(a) U-user-sim-ind

2	4	3	1	0
2	4	3	0	1
3	0	1	4	2
2	4	1	0	3
3	0	2	1	4

(b) U-loc-sim-ind

3	1	2	0
2	0	3	1
3	1	0	2
2	0	1	3

(c) U-act-sim-ind

Figure 3.10: Sorted similarity indices matrices of CB method

Note that since in the $U-user$ and $U-loc$ matrices, all values with a significant value are gathered in one column. Since cosine distance works by comparing "the angle" between the vectors and does not take into account the magnitude, the similarities between users and locations will be zero. For the sake of this example, we used euclidean distance instead of cosine distance and got the similarity matrices according to that. This problem is not likely to happen in a real-world data set, where there will be hundreds of columns.

This time we will try to predict the rating value of the entry $e_{(0,1,0)}$ of the tensor. The values we get are as follows:

$$P^u_{User} = \frac{2.791 * 0.19}{0.19} = 2.791$$

$$P^u_{Loc} = \frac{1.0 * 0.0019}{0.019} = 1.0, P^u_{Act} = undefined$$

$$\hat{T}_{CB}(0,1,0) = \frac{2.791 + 1.0}{2} = 1.8955$$

For P^u_{User}, we select user 1 which is the most similar user. For P^u_{Loc}, we have to select location 0 since there is no other rated location of user 0. P^u_{Act} is undefined. At the end, we predict the rating as 1.8955.

3.5.4 Hybrid Method

For Hybrid method, we have similar matrices as in other methods up to now. In our constructed 2-D matrix, we combine similarity values from the values we obtained from the original tensor. This is the formulation shown in Equation 3.7. The corresponding matrices of the process are shown in Figures 3.11 to 3.14.

0	0.4	0	0.6	0	0	0	0	0	0	0	0	0	0
0.67	0	0	0.33	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.75	0.25	0	0	0	0	0	0	0	0	0	0	0	0
1.5	0	0	0	0	0	0.5	0.5	0	0	0	0	0	0
2.3	0	0	0	0	0	0.57	0	0.43	0	0	0	0	0
0	2	0	0	0.095	0.19	0	0.62	0.095	0	0	0	0	0
0	0	1.4	0	0.12	0	0.81	0	0.062	0	0	0	0	0
0	0	0	1.5	0	0.5	0.33	0.17	0	0	0	0	0	0
1.7	2.8	3.4	0	1	0	1	0	2.6	0	0.28	0.32	0.2	0.2
4.3	0	0	0	2.4	2.8	2.4	0	1.7	0.44	0	0.38	0	0.19
1.5	3.1	1	0	1	0	1	3.1	3.1	0.35	0.26	0	0.17	0.22
0	2.8	1	1	1	0	1	2.8	1	0.42	0	0.33	0	0.25
0	1	2.9	1	1	0	1	2.9	1	0.31	0.19	0.31	0.19	0

Figure 3.11: Constructed matrix of Hybrid method

0.76	0.04	0.48
0.16	0.8	0.39
0.31	0.16	0.54
0.38	0.32	0.38
0.045	0.32	0.37

(a) U-user

0.76	0.04	0.48
0.16	0.8	0.39
0.31	0.16	0.54
0.38	0.32	0.38
0.045	0.32	0.37

(b) U-loc

0.036	0.033	0.023
0.018	0.068	0.027
$1.2e-17$	$1.9e-17$	$2e-17$
0.0053	0.068	0.039

(c) V-loc

0.096	0.0098	0.062
0.042	0.018	0.034
0.021	0.013	0.053
0.032	0.016	0.025
0.012	0.0042	0.037

(d) V-user

0.072	0.17	0.27
0.16	0.36	0.11
0.032	0.21	0.28
0.61	0.38	0.39
0.13	0.035	0.43

(e) V-loc

0.0062	0.67	0.37
0.052	0.35	0.46
0.73	0.26	0.37
0.17	0.1	0.081

(f) V-act

Figure 3.12: Reduced-rank U and V matrices of Hybrid method

0.58	0.49	0.14	0.13	0
0.58	0.33	0.18	0.084	0
0.33	0.16	0.13	0.071	$1.1e-16$
0.18	0.16	0.14	0.071	$1.1e-16$
0.49	0.16	0.16	0.084	$1.1e-16$

(a) U-user-sim

0.4	0.35	0.12	0.03	$2.2e-16$
0.84	0.79	0.25	0.12	0
0.25	0.24	0.21	0.03	0
0.79	0.35	0.21	0.0015	$1.1e-16$
0.84	0.4	0.24	0.0015	0

(b) U-loc-sim

0.22	0.14	0.059	$1.1e-16$
0.14	0.092	0.024	$1.1e-16$
0.097	0.092	0.059	0
0.22	0.097	0.024	0

(c) U-act-sim

0.22	0.19	0.044	0.035	0
0.18	0.13	0.035	0.00066	$1.1e-16$
0.19	0.13	0.13	0.0088	0
0.19	0.13	0.044	0.00066	0
0.22	0.19	0.18	0.0088	0

(d) V-user-sim

0.23	0.2	0.11	0.011	0
0.56	0.23	0.23	0.17	$1.1e-16$
0.28	0.23	0.16	0.011	0
0.3	0.28	0.2	0.17	0
0.56	0.3	0.16	0.11	$1.1e-16$

(e) V-loc-sim

0.52	0.38	0.09	0
0.4	0.33	0.09	0
0.52	0.4	0.02	0
0.38	0.33	0.02	0

(f) V-act-sim

Figure 3.13: Sorted similarity matrices of Hybrid

1	4	3	2	0
0	2	3	4	1
1	4	0	3	2
1	4	0	2	3
0	2	3	1	4

(a) U-user-sim-ind

4	3	1	2	0
4	3	2	0	1
1	4	3	0	2
1	0	2	4	3
1	0	2	3	4

(b) U-loc-sim-ind

3	1	2	0
0	2	3	1
3	1	0	2
0	2	1	3

(c) U-act-sim-ind

4	2	3	1	0
4	2	0	3	1
0	3	1	4	2
4	2	0	1	3
0	3	1	2	4

(d) V-user-sim-ind

1	3	4	2	0
4	0	2	3	1
3	1	4	0	2
4	2	0	1	3
1	3	2	0	4

(e) V-loc-sim-ind

2	3	1	0
2	3	0	1
0	1	3	2
0	1	2	3

(f) V-act-sim-ind

Figure 3.14: Sorted similarity indices matrices of Hybrid

If we want to calculate the prediction of the entry $e_{(0,1,0)}$, we will follow the steps we followed for the other methods again. For brevity, we only give the calculated values this time. Note that values coming from activity dimension are zero again, this is because the user do no have any other ratings with the same location and a different activity.

$$P^u_{User} = 2.792, P^v_{User} = 1.693, P^u_{Loc} = 1.0$$

$$P^v_{Loc} = 1.0, P^u Act = undefined, P^v_{Act} = undefined$$

$$\hat{T}_{Hyb}(0,1,0) = \frac{2.792 + 1.693 + 1.0 + 1.0}{4} = 1.621$$

CHAPTER 4

SIMILARITY ESTIMATION BASED RECOMMENDATION

In this section we present a new prediction method which does not use Singular Value Decomposition and makes calculation using the similarity matrices directly. From this point on, we will call this method as *SimPred*.

Singular Value Decomposition has two uses in general. First, it is used to find latent indexing among the data. Secondly, it is used to reduce the rank of the data while losing some information. With a robust similarity calculation technique, the first use is redundant. Moreover, if the performance of the algorithm is acceptable for big data, the second use is unneeded as well. Plus, the cost of applying SVD is avoided.

In addition, we saw that SVD did not produce significant reductions without losing the information greatly. For example in our middle-size data set, rank of 2-D matrix before reduction was 4000. In order to reduce this dimension 482, we have to lose 50% of the data. If we want to keep higher percentage of data, say 80% for example, then dimensionality was reduced to 1019 only. Even in a middle-size test data set like ours, this dimension is not satisfactory. For real data sets, the problem is of course more severe. Moreover, we still have to do the computationally costly SVD calculations.

In order to overcome these problems, we propose an algorithm which uses directly the similarity matrices calculated from the check-ins. The calculation technique is described in Section 1.1. We then sort these similarity matrices in order get the most similar items. Then using these sorted matrices, we make prediction over using the most similar items. Note that we do not get three different values for 3 different dimensions. The algorithm is shown in Algorithm 2.

```
Data: T : tensor,
(UU, LL, AA) : sorted similarity matrices,
(UU_ind, LL_ind, AA_ind): sorted indices of sim matrices
Result: $\hat{t}$ : prediction
predictions = [], weights = [];
$\hat{t} = 0$;
curu = 0, curl = 0, cura = 0;
while size of predictions < neighborhood size do
    which = max(UU[u][curu], LL[l][curl], AA[a][cura]);
    if which == 0 then
        i = (UU_ind[u][curu], l, a); curu++
    end
    if which == 1 then
        i = (u, LL_ind[l][curl], a); curl++
    end
    if which == 2 then
        i = (u, l, AA_ind[a][cura]); cura++;
    end
    if T(i) != 0 then
        add T(i) to predictions add similarity to weights
    end
end
for i=0 to neighborhood size do
    $\hat{t} = \hat{t} + predictions[i] * weights[i]$
end
$\hat{t} = \hat{t} / \sum weights$
```

Algorithm 2: SimPred Algorithm

In general, time complexity of applying SVD on an nxn matrix is given as $O(n^3)$ [34]. It is the bottleneck of SVD-based prediction methods. In SimPred, since we do not use SVD, we reduce this complexity to $O(n * log(n))$.

4.1 Running Example

In this section we give a small running example using the tensor shown in Figure 3.1. We will try to predict the value of entry $e_{(0,1,0)}$.

The algorithm first checks the similar items in all dimensions and tries them in the order of their similarity. The algorithm checks first row of location-location similarity matrix, and second rows of user-user and activity-activity similarity matrices. We find out that activity 3 has the most similarity with 0.67. But since the value of $T(0,1,3)$ is zero, we ignore it. We select the next item with the most similarity, which is location 2. We check the value of $T(0,3,0)$ and find that it is zero as well. In this way we test location 4, activity 1 and user 2. For user 2, we see that $T(2,1,0)$ is non-zero. Since, our neighborhood size is 1, we stop the algorithm here and calculate the prediction as follows:

$$\hat{T}_{Sim}(0,1,0) = \frac{T(2,1,0) * U_{User}(0,2)}{U_{User}(0,2)} = \frac{2.38629 * 0.32}{0.32} = 2.38629$$

CHAPTER 5

EVALUATION

In this chapter we present the evaluation results of the proposed methods. We first describe our data sets and present the results gathered in them. Then we make a parameter optimization analysis. Then we present the evaluation results of improved SVD-based method and similarity based method in different sections.

5.1 Data Sets

In order to evaluate our methods, we used two different data sets. First data set was obtained from Gowalla and the second from Brightkite. Both are location-based applications which are currently discontinued [10].

The data sets consists of a number of check-ins gathered over a period of time. Each check-in has user ID, date and time, altitude, longitude and a location ID fields [10]. In addition, we have found the corresponding locations from Foursquare and used the category of these locations as the activity information.

For Gowalla data set, we extracted 3 different-size subsets from the data. Small data set contains 309 users, 132 locations, 82 activities. Medium data set contains 2136 users, 1603 locations, 267 activities. Finally the large data set contains 11216 users, 7688 locations and 407 activities.

For Brightkite data set, two different sized data sets were obtained. Small set contains 517 users, 318 locations and 130 activities whereas medium data set contains 2126 users, 1476 locations and 305 activities.

5.2 Evaluation Metrics

In order to evaluate the performance of the proposed methods, we have used the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) metrics for assessing the performance of our proposed methods. MAE is used to assess the average error of the methods and RMSE gives insight about the spread of the error. A discussion about MAE and RMSE is given in [5], [18] and [36]. Definition of MAE and $RMSE$ is given in Equations 5.1 and 5.2 where O and E are sets of output and expected values respectively. K-fold validation is applied on the data with k=10 folds throughout all the experiments. We present the averages obtained from these 10 folds.

$$MAE(O,E) = \frac{1}{n}\sum_{i}^{n} |o_i - e_i| \tag{5.1}$$

$$RMSE(O,E) = \sqrt{\frac{\sum_i^n (o_i - e_i)^2}{n}} \tag{5.2}$$

5.3 Analysis for Parameter Optimization

In Figures 5.1 to 5.4, we present the impact of neighborhood sizes on the performance of methods. We have determined 7 as the testing neighborhood size although in 1 or 2 cases of small data set 6 seems a better choice in terms of MAE. For all experiments, we use the neighborhood size of 7.

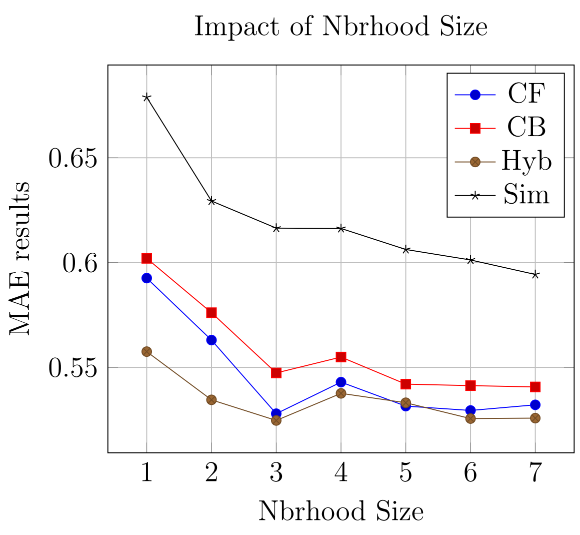

Figure 5.1: Impact of Neighborhood Size on MAE in Small Gowalla Data Set

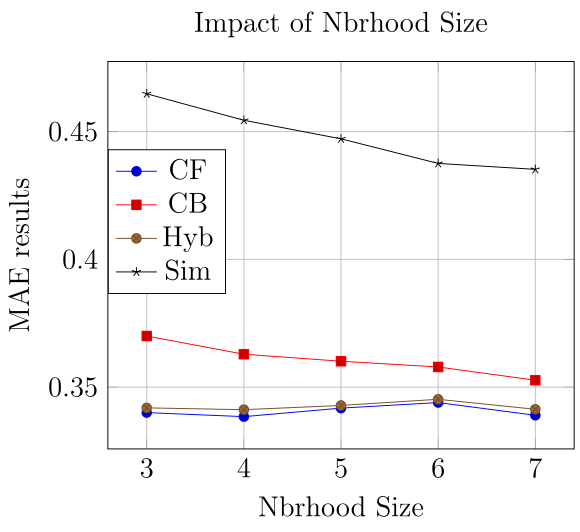

Figure 5.2: Impact of Neighborhood Size on MAE in Medium Gowalla Data Set

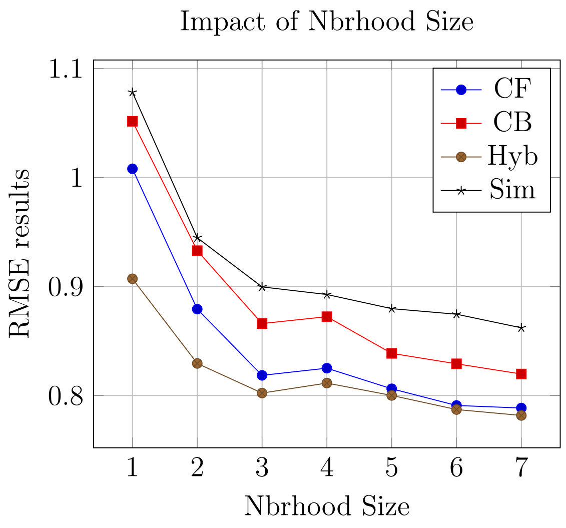

Figure 5.3: Impact of Neighborhood Size on RMSE in Small Gowalla Data Set

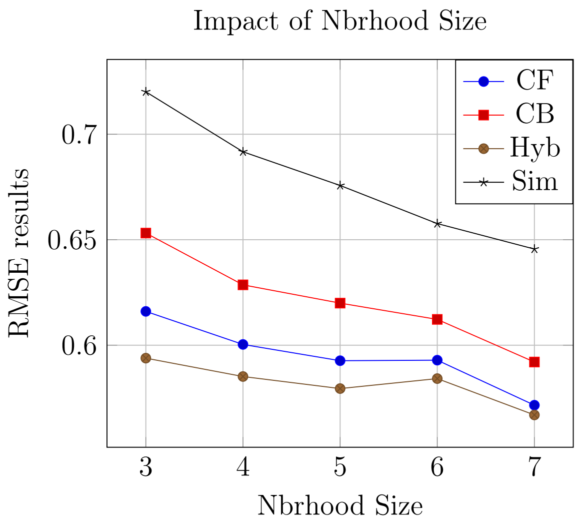

Figure 5.4: Impact of Neighborhood Size on RMSE in Medium Gowalla Data Set

5.4 Experimental Evaluation of the Improved SVD-Based Recommendation Method

In this section we present the evaluation results for the proposed SVD-based methods. We will present the results for graph-based similarity calculation method, weighted average calculation and overall comparison in different subsections.

5.4.1 Graph-Based Similarity Calculation

First we present the impact of our new similarity calculation method on SVD-based methods. Figures 5.5 to 5.8 show the impact of similarity calculation methods in Gowalla and Brightkite data sets. Figures 5.9 to 5.10 show the impact in time required for preparation.

In Figures 5.9 and 5.10, we can see that proposed similarity methods lead to shorter preparation times, This is because we have denser matrices with the original methods. As seen in the figures, this effect is most prominent in CB method. This is because in CB method, the only information used is similarity information. In Hybrid method, the information coming from original tensor dominates the constructed 2-D matrix. This effect gets more dramatic for the big Gowalla and medium Brightkite data sets that after waiting for several hours for preparation, we stopped the experiment. For the big Gowalla data set we waited for 8 hours and for the medium Brightkite data set for 40 hours.

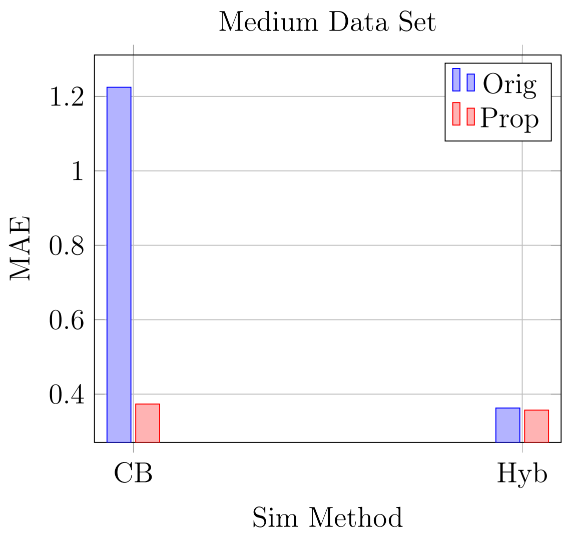

Figure 5.5: MAE of Similarity Calculation Methods in Gowalla Data Set

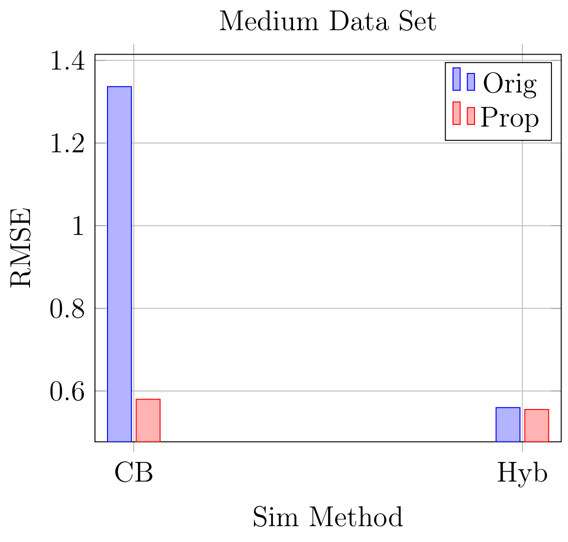

Figure 5.6: RMSE of Similarity Calculation Methods in Gowalla Data Set

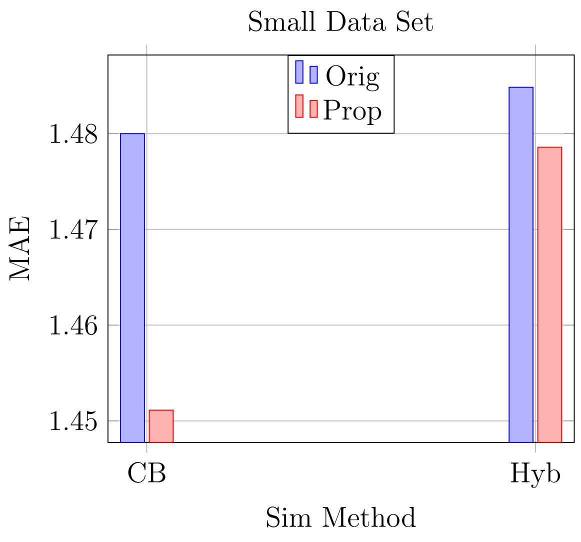

Figure 5.7: MAE of Similarity Calculation Methods in Brightkite Data Set

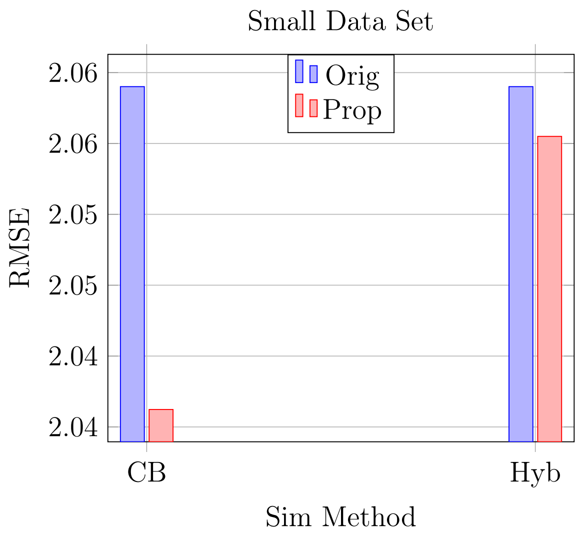

Figure 5.8: RMSE of Similarity Calculation Methods in Brightkite Data Set

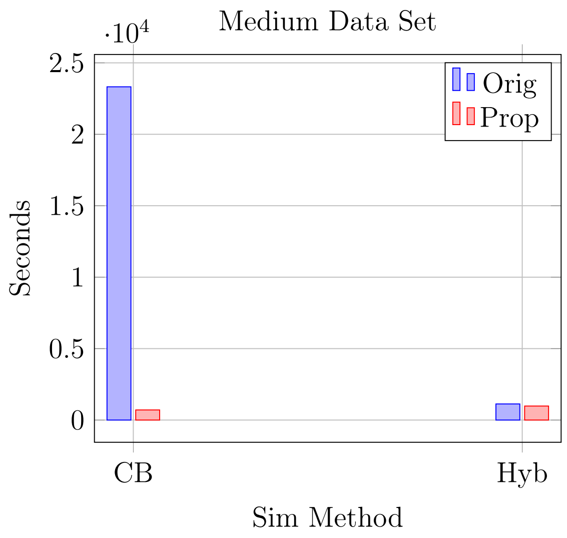

Figure 5.9: Preparation Times of SVD-based methods using different Similarity Matrices in Gowalla Data Set

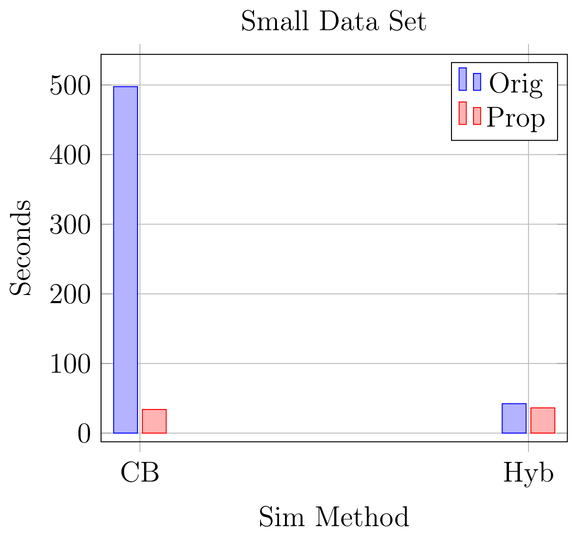

Figure 5.10: Preparation Times of SVD-based methods using different Similarity Matrices in Brightkite Data Set

5.4.2 Weighted Average Calculation

We then present the MAE and RMSE results of using weighted average and deviation average in SVD prediction methods. Note that the proposed similarity methods are used in these experiments. As can be seen in Figures 5.11 to 5.14 weighted average produces better MAE results. In Figures 5.15 and 5.18, we see that new similarity calculation gives increased RMSE only in Figure 5.17. This means we have a slightly bigger spread in the error with the new calculation technique in that figure. This is not a problem since this increase is lower than the reduction in MAE.

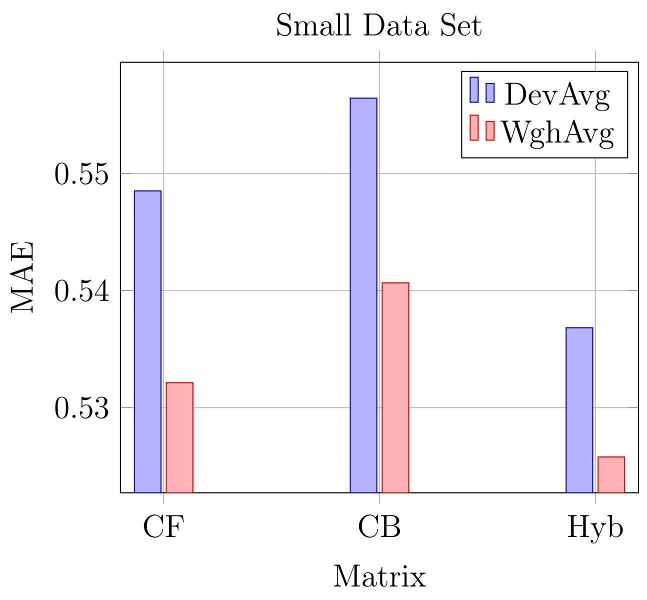

Figure 5.11: MAE of Average Calculation Methods in Small Gowalla Data Set

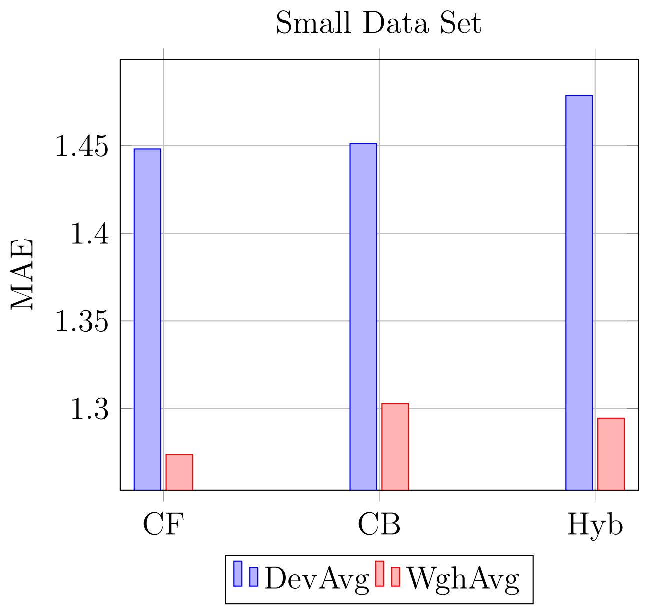

Figure 5.12: MAE of Average Calculation Methods in Small Brightkite Data Set

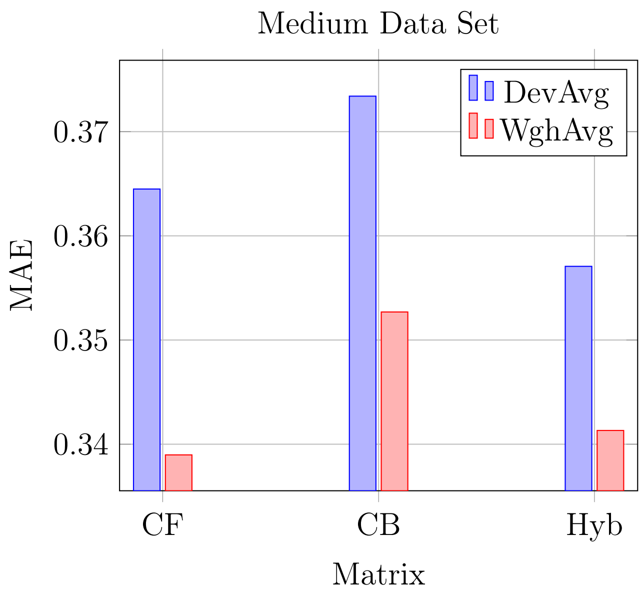

Figure 5.13: MAE of Average Calculation Methods in Medium Gowalla Data Set

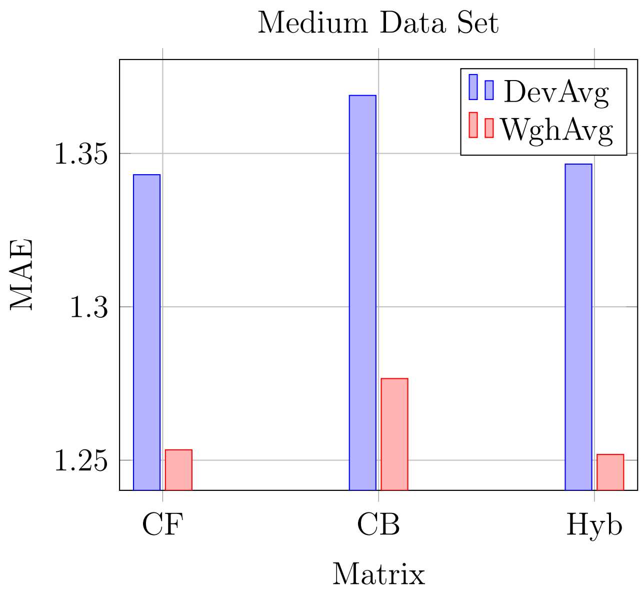

Figure 5.14: MAE of Average Calculation Methods in Medium Brightkite Data Set

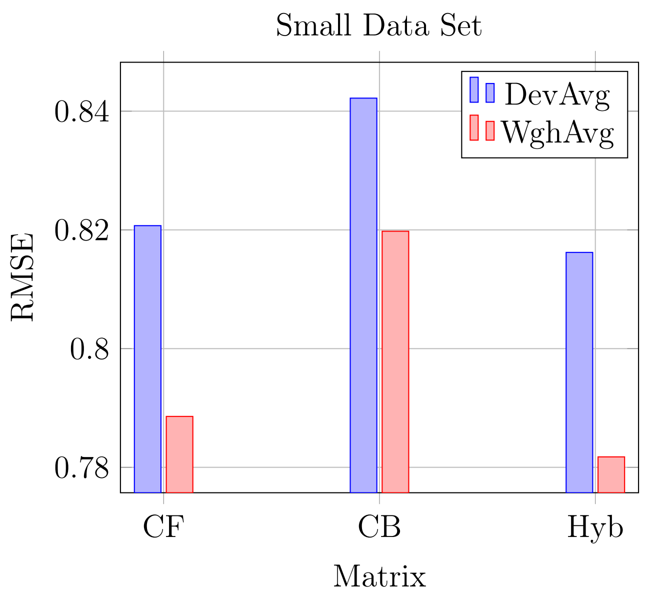

Figure 5.15: RMSE of Average Calculation Methods in Small Gowalla Data Set

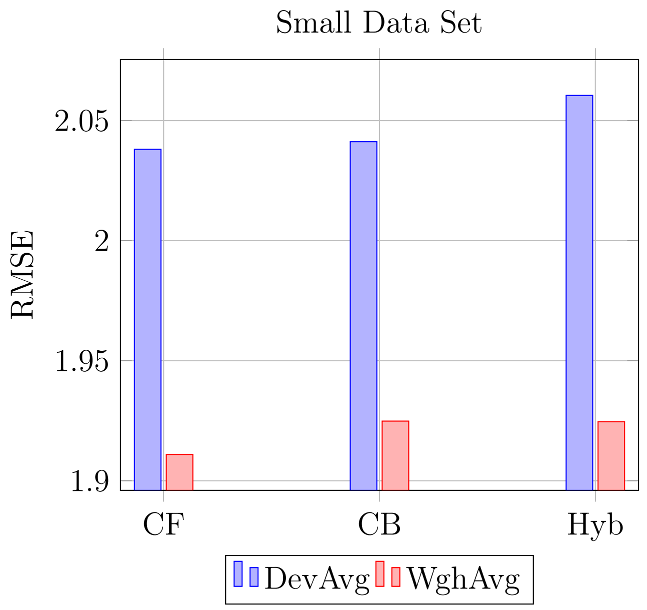

Figure 5.16: RMSE of Average Calculation Methods in Small Brightkite Data Set

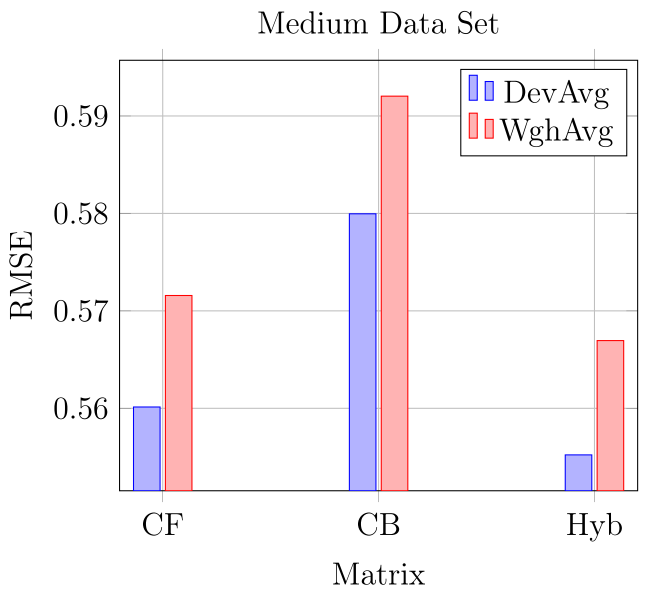

Figure 5.17: RMSE of Average Calculation Methods in Medium Gowalla Data Set

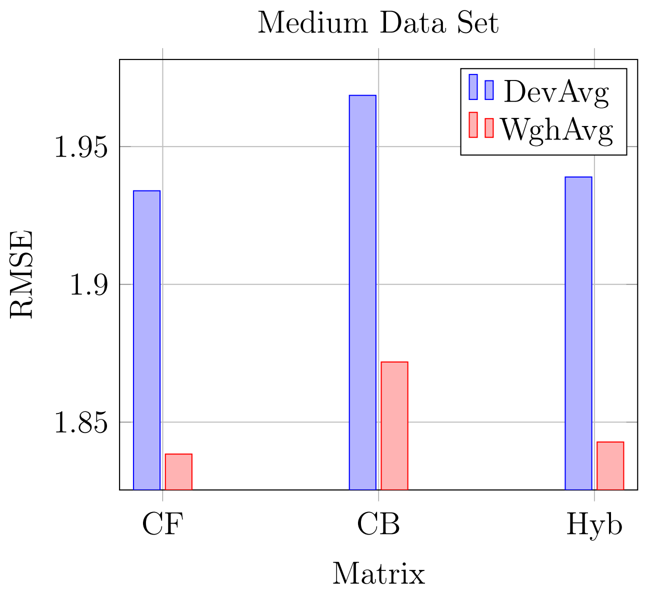

Figure 5.18: RMSE of Average Calculation Methods in Medium Brightkite Data Set

5.4.3 Overall Comparison

In Figures 5.19 to 5.24 we compare our results with the results in [30]. Because of the very long running time of the original method in big Gowalla and medium

Brightkite data sets, we could not get values of original methods for comparing with our values. It is seen that both in small and medium sized data sets, proposed methods produce better MAE results. In addition, except a small increase in RMSE result of *Hyb_ Pr* in Figure 5.24, RMSE results are lower as well.

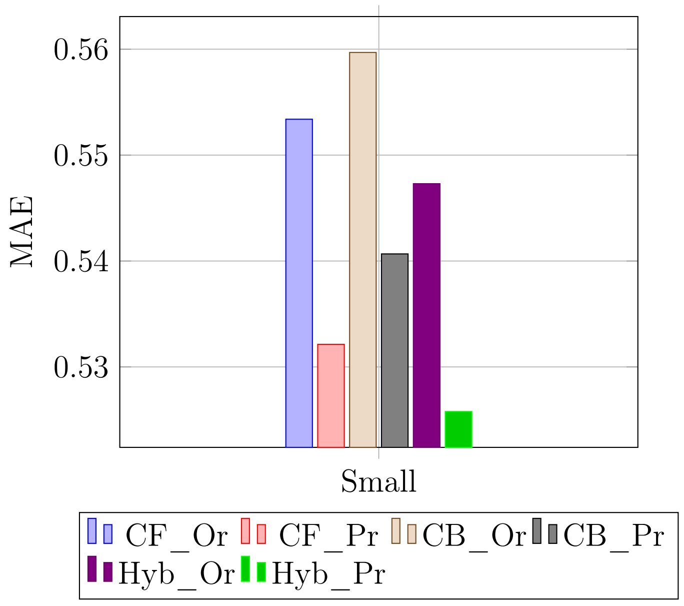

Figure 5.19: Overall Comparison of MAE Results in Small Gowalla Data Set

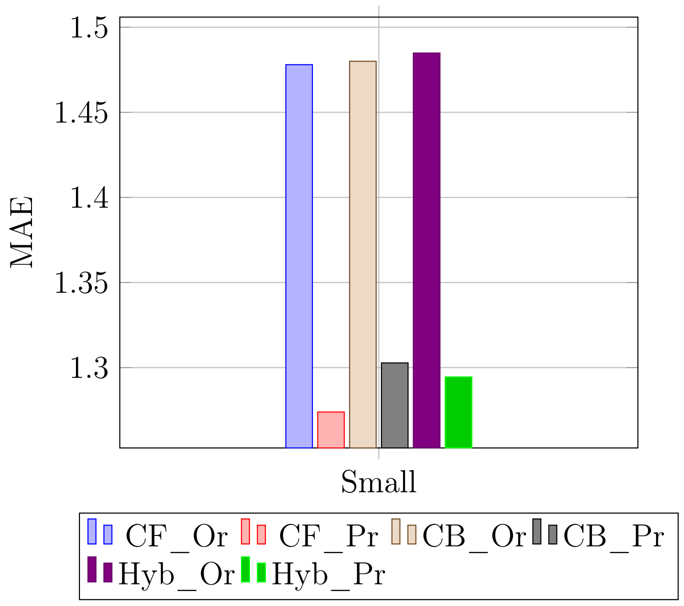

Figure 5.20: Overall Comparison of MAE Results in Small Brightkite Data Set

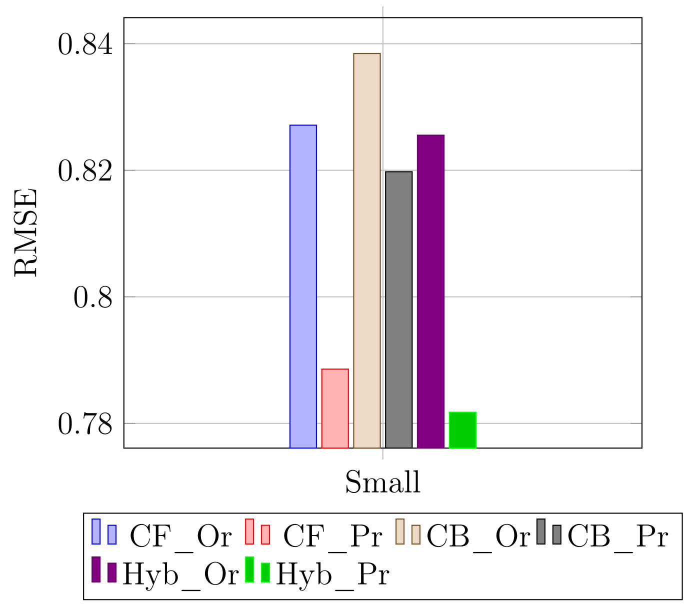

Figure 5.21: Overall Comparison of RMSE Results in Small Gowalla Data Set

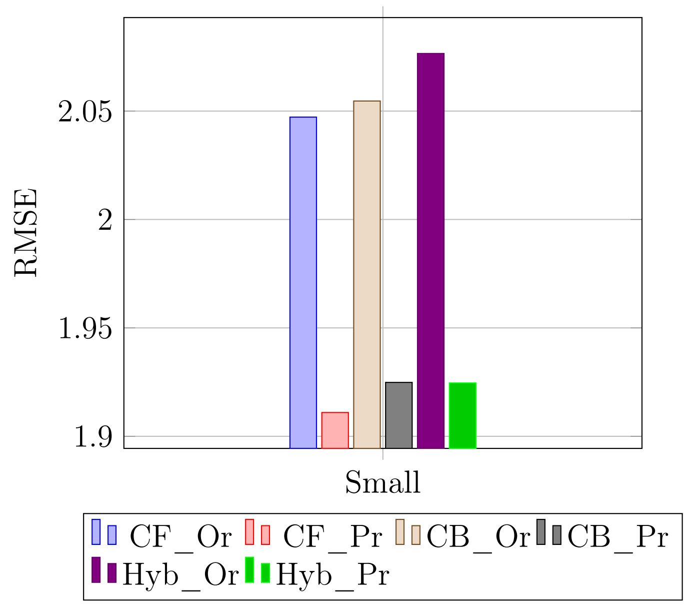

Figure 5.22: Overall Comparison of RMSE Results in Small Brightkite Data Set

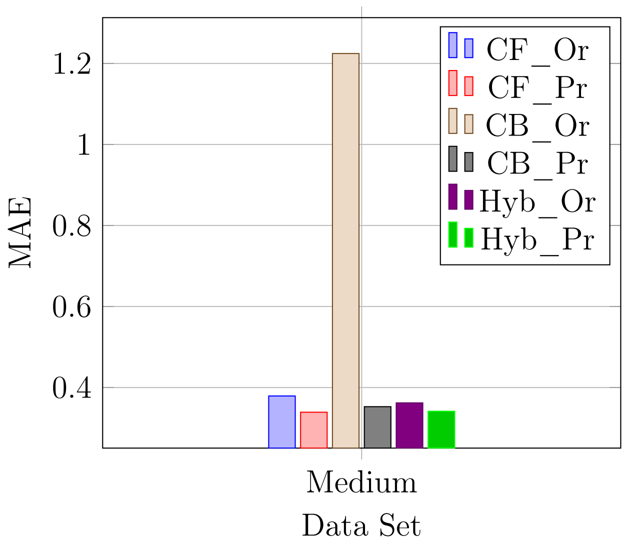

Figure 5.23: Overall Comparison of MAE Results in Medium Gowalla Data Set

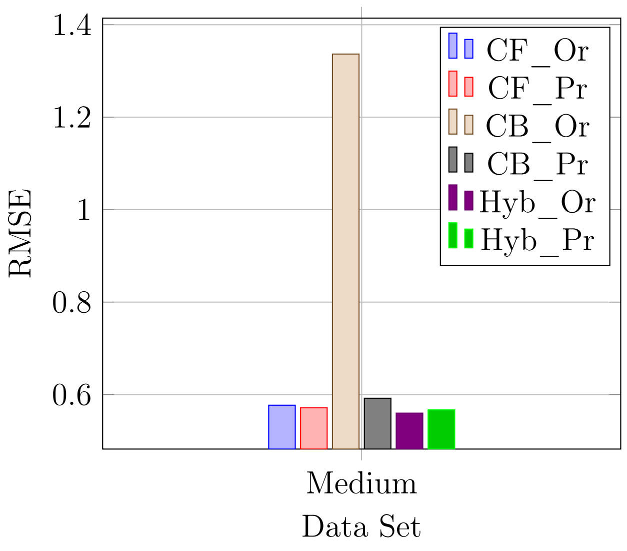

Figure 5.24: Overall Comparison of RMSE Results in Medium Gowalla Data Set

5.5 Experimental Evaluation of the Similarity-Based Recommendation Method

In this section we present the results related to the similarity based recommendation method, SimPred.

From Figure 5.25 to 5.28, we give performance comparisons of SimPred method with our SVD-based methods. Then from Figure 5.29 to 5.30, preparation time comparison of the proposed methods are given. It is seen that in all sizes and methods, SimPred improves run-time efficiency with an acceptable trade-off in performance.

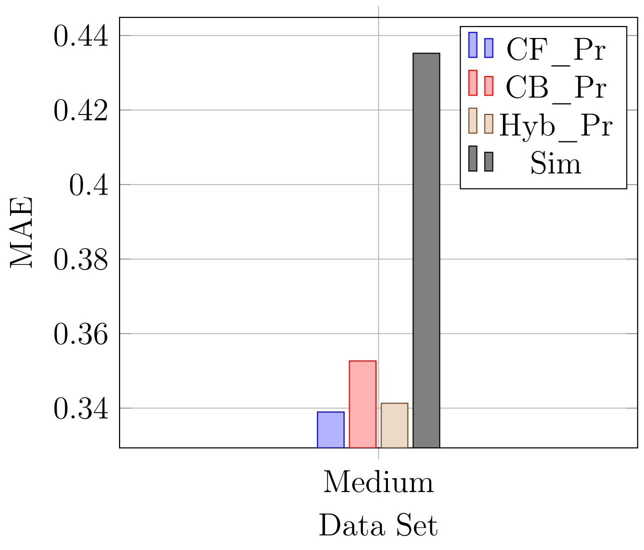

Figure 5.25: MAE Comparison of SimPred method in Medium Gowalla Data Set

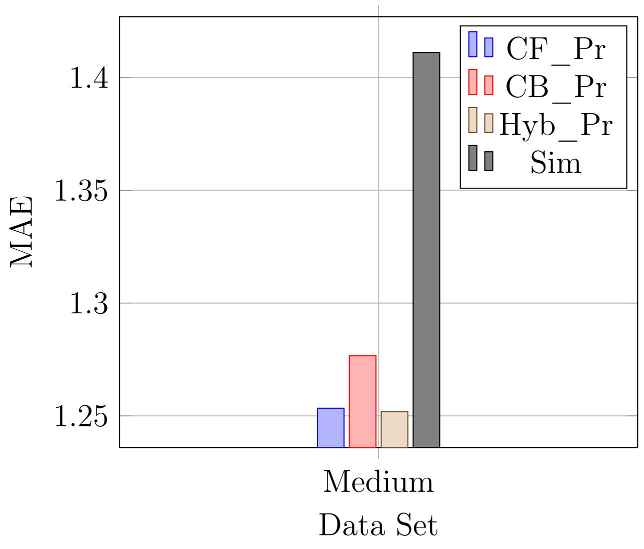

Figure 5.26: MAE Comparison of SimPred method in Medium Brightkite Data Set

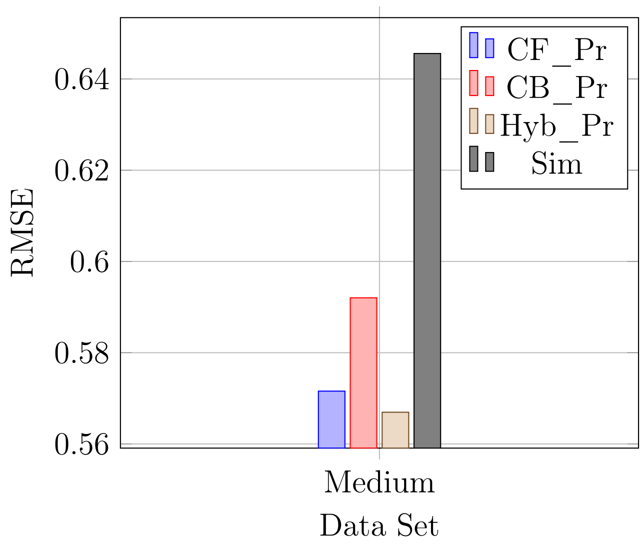

Figure 5.27: RMSE comparison of SimPred method in Medium Gowalla Data Set

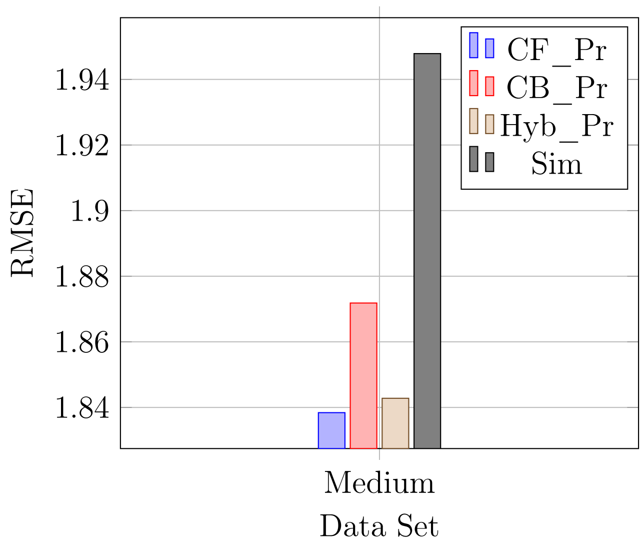

Figure 5.28: RMSE comparison of SimPred method in Medium Brightkite Data Set

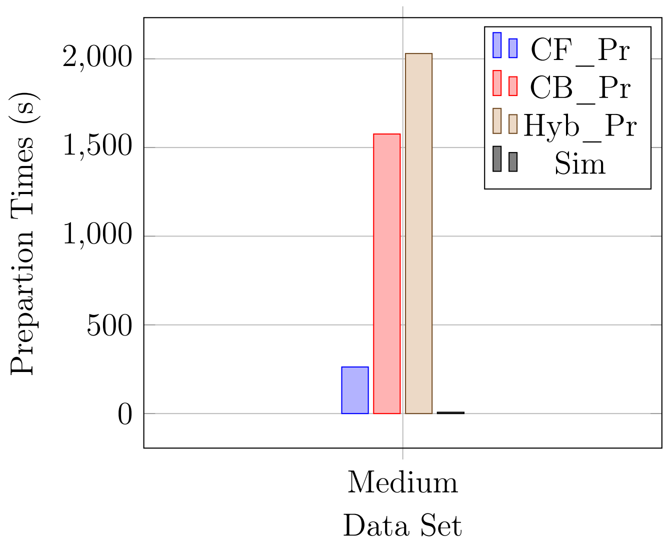

Figure 5.29: Preparation Time Comparison of Proposed Methods in Medium Gowalla Data Set

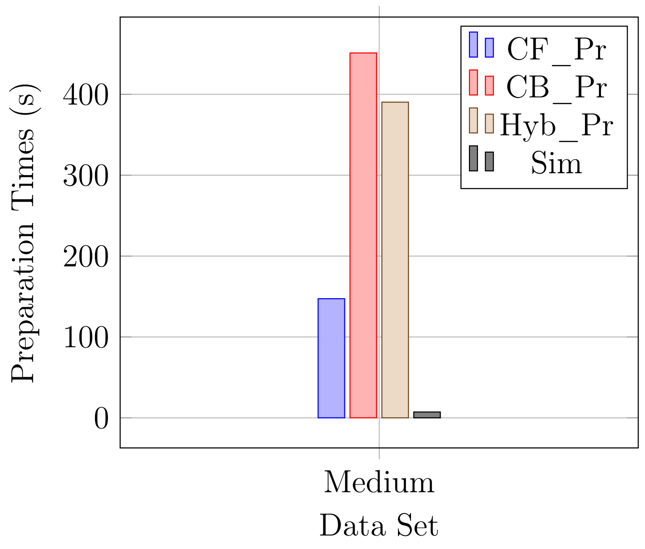

Figure 5.30: Preparation Time Comparison of Proposed Methods in Medium Brightkite Data Set

In Table 5.1 preparation time and performance comparisons of SimPred method can be seen quantitatively against the proposed hybrid SVD-based method. It is seen that the enhancement in preparation time is significantly better than the loss in MAE and RMSE. Since complexity of SimPred method is $O(n * log(n))$ and it is suitable for use with sparse matrices, it can be used with very big data.

Table5.1: Rate of SimPred Results Against Hybrid SVD-Based Method

	MAE	**RMSE**	**Prep Time**
Medium Gowalla Data Set	1.27522	1.13873	0.00363
Medium Brightkite Data Set	1.1271	1.057	0.01845

CHAPTER 6

CONCLUSION AND FINAL REMARKS

In this thesis we have presented an improvement to the algorithm described in [30] and a new method (SimPred) which has acceptable results and dramatically better efficiency in large data sets.

As a summary, we improved [30] with a more robust and scalable similarity calculation technique and a better average calculation method. We also discussed that SVD-based methods are not suitable as real world applications since their complexity in offline phase is too high. In order to solve this, we proposed a brand new algorithm which reduces the preparation time dramatically with an acceptable trade-off in rating performance.

Because of the nature of our data sets, a location could be associated with only one category in our setup. A setup with locations having more than one activity associated with them can give different and possibly better results.

One other aspect we noticed in the evaluation phase was the check-in habits of location based social network users. We noticed that most users tend to make a check-in in a place only once. Even if they come later again, apparently they do not make a second check-in. Because of this, their relationships with locations become a binary relation and rating extraction loses its meaning. We believe for those people, a binary recommender technique can be explored.

SimPred may also be improved further by incorporating other data such as trust between users. A similar approach to [22] can be followed. As an example, number of likes between users in Facebook can be used for this purpose.

Unfortunately our data was not suitable for this.

Some popular techniques on graph data structures such as random walk with restart algorithm [24] may be applied on the dimensions to get enhanced similarity results. [33] showed that there are fast applications of them.

REFERENCES

[1] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems (TOIS)*, 23(1):103–145, 2005.

[2] G. Adomavicius and A. Tuzhilin. Multidimensional recommender systems: a data warehousing approach. In *Electronic commerce*, pages 180–192. Springer, 2001.

[3] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[4] C. C. Aggarwal, J. L. Wolf, K.-L. Wu, and P. S. Yu. Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 201–212. ACM, 1999.

[5] J. S. Armstrong. Evaluating forecasting methods. In *Principles of forecasting*, pages 443–472. Springer, 2001.

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

[7] R. Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[8] R. Burke. Hybrid web recommender systems. In *The adaptive web*, pages 377–408. Springer, 2007.

[9] S. Chaudhuri and U. Dayal. An overview of data warehousing and olap technology. *ACM Sigmod record*, 26(1):65–74, 1997.

[10] E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1082–1090. ACM, 2011.

[11] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[12] S. Gauch, M. Speretta, A. Chandramouli, and A. Micarelli. User profiles for personalized information access. In *The adaptive web*, pages 54–89. Springer, 2007.

[13] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[14] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237. ACM, 1999.

[15] Q. Huang and Y. Liu. On geo-social network services. In *Geoinformatics, 2009 17th International Conference on*, pages 1–6. IEEE, 2009.

[16] Z. Huang, H. Chen, and D. Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):116–142, 2004.

[17] Z. Huang, W. Chung, T.-H. Ong, and H. Chen. A graph-based recommender system for digital library. In *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 65–73. ACM, 2002.

[18] R. J. Hyndman and A. B. Koehler. Another look at measures of forecast accuracy. *International journal of forecasting*, 22(4):679–688, 2006.

[19] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender systems: an introduction.* Cambridge University Press, 2010.

[20] S. Lee, J. Yang, and S.-Y. Park. Discovery of hidden similarity on collaborative filtering to overcome sparsity problem. In *Discovery Science*, pages 396–402. Springer, 2004.

[21] G. Linden, B. Smith, and J. York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[22] P. Massa and P. Avesani. Trust-aware collaborative filtering for recommender systems. In *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, pages 492–508. Springer, 2004.

[23] K. Miyahara and M. J. Pazzani. Collaborative filtering with the simple bayesian classifier. In *PRICAI 2000 Topics in Artificial Intelligence*, pages 679–689. Springer, 2000.

[24] J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu. Automatic multimedia cross-modal correlation discovery. In *Proceedings of the tenth ACM*

SIGKDD international conference on Knowledge discovery and data mining, pages 653–658. ACM, 2004.

[25] J. Parsons, P. Ralph, and K. Gallagher. Using viewing time to infer user preference in recommender systems. 2004.

[26] M. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine learning*, 27(3):313–331, 1997.

[27] M. J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5-6):393–408, 1999.

[28] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, DTIC Document, 2000.

[29] 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*, pages 285–295. ACM, 2001.

[30] M. Sattari. A Hybrid Geo-Activity Recommendation System Using Advanced Feature Combination and Semantic Activity Similarity. Master's thesis, Middle East Technical University, Ankara, Turkey, 2013.

[31] M. Sattari, M. Manguoglu, I. H. Toroslu, P. Symeonidis, P. Senkul, and Y. Manolopoulos. Geo-activity recommendations by using improved feature combination. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 996–1003. ACM, 2012.

[32] J. B. Schafer, J. A. Konstan, and J. Riedl. E-commerce recommendation applications. In *Applications of Data Mining to Electronic Commerce*, pages 115–153. Springer, 2001.

[33] H. Tong, C. Faloutsos, and J.-Y. Pan. Fast random walk with restart and its applications. 2006.

[34] L. N. Trefethen and D. Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.

[35] Z. Wang, Y. Tan, and M. Zhang. Graph-based recommendation on social networks. In *Web Conference (APWEB), 2010 12th International Asia-Pacific*, pages 116–122. IEEE, 2010.

[36] C. J. Willmott and K. Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate Research*, 30(1):79, 2005.

[37] S. Wright and J. Nocedal. *Numerical optimization*, volume 2. Springer New York, 1999.

[38] Z. Yuan, Y. Jiang, and G. Gidófalvi. Geographical and temporal similarity measurement in location-based social networks. In *Proceedings of the Second ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems*, pages 30–34. ACM, 2013.

[39] V. W. Zheng, B. Cao, Y. Zheng, X. Xie, and Q. Yang. Collaborative filtering meets mobile recommendation: A user-centered approach. In *AAAI*, volume 10, pages 236–241, 2010.

[40] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang. Collaborative location and activity recommendations with gps history data. In *Proceedings of the 19th international conference on World wide web*, pages 1029–1038. ACM, 2010.

[41] Y. Zheng, Y. Chen, X. Xie, and W.-Y. Ma. Geolife2. 0: a location-based social networking service. In *Mobile Data Management: Systems, Services and Middleware, 2009. MDM'09. Tenth International Conference on*, pages 357–358. IEEE, 2009.