A HYBRID GEO-ACTIVITY RECOMMENDATION SYSTEM USING ADVANCED FEATURE COMBINATION AND SEMANTIC ACTIVITY SIMILARITY

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

BY

MASOUD SATTARI

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

SEPTEMBER 2013

Approval of the thesis:

A HYBRID GEO-ACTIVITY RECOMMENDATION SYSTEM USING ADVANCED FEATURE COMBINATION AND SEMANTIC ACTIVITY SIMILARITY

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

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı _____
Head of Department, **Computer Engineering**

Prof. Dr. İsmail Hakkı Toroslu _____
Supervisor, **Computer Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Faruk Polat _____
Computer Engineering Dept., METU

Prof. Dr. İsmail Hakkı Toroslu _____
Computer Engineering Dept., METU

Assoc. Prof. Dr. Pınar Karagöz _____
Computer Engineering Dept., METU

Assoc. Prof. Dr. Murat Manguoğlu _____
Computer Engineering Dept., METU

Dr. Güven Fidan _____
CEO, ArGeDor Information Technologies

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: MASOUD SATTARI

Signature :

ABSTRACT

A HYBRID GEO-ACTIVITY RECOMMENDATION SYSTEM USING ADVANCED FEATURE COMBINATION AND SEMANTIC ACTIVITY SIMILARITY

Sattari, Masoud
M.Sc., Department of Computer Engineering
Supervisor: Prof. Dr. İsmail Hakkı TOROSLU

September 2013, 67 pages

With booming technology of smart mobile phones and satellite-assisted positioning systems, new demands for applications in this field are emerging. One of these requirements that most of the users are interested in, is activity recommendation based on location (GPS) data, which is specially used to guide tourists and unfamiliar individuals in tourist-attracting cities. Therefore, location-based social networks (LBSN) and location-based recommendations have emerged as interesting research topics in literature.

To deal with drawbacks of individual recommendation techniques, various hybrid systems have been proposed. In this thesis, we aim to promote the accuracy of Geo-activity recommendation system which is generally presented by 2-D Location-Activity rating matrix. We accomplish this task by combining data from different resources and exploiting well-known advantages of Singular Value Decomposition (SVD) method. This method has the functionality of uncovering latent relation within data and reducing its rank. Furthermore, we extend the proposed method, to reduce a recommendation of form User-Location-Activity in 3-D rating Tensor to a problem of recommendation in 2-D matrix. We accomplish this reduction by means of High Order Singular Value Decomposition (HOSVD) as well as merging various 2-D matrices to construct an integrated matrix. In addition, activity-activity similarity matrix is extracted from a semantic structure in our proposed method.

Keywords: Recommender Systems, Collaborative Filtering, Singular Value Decomposition, Feature Combination, Semantic Similarity, High Order Singular Value Decomposition

ÖZ

İLERİ ÖZELLİK BİRLEŞTİRME VE ANLAMSAL EYLEM BENZERLİĞİ KULLANAN MELEZ COĞRAFİ EYLEM ÖNERİ SİSTEMİ

Sattari, Masoud
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. İsmail Hakkı TOROSLU

Eylül 2013, 67 sayfa

Taşınabilir telefonlar ve uydu destekli konumlandırma sistemlerinin gelişmesi ile, bu alandaki uygulamalar için yeni ihtiyaçlar ortaya çıkmaktadır. Kullanıcıları en çok ilgilendirenlerden gereksinimlerden birisi, özellikle turistlerin ve yabancı bireylerin şehirlerde yönlendirilmesi için kullanılan, konum (İng., GPS) verisi tabanlı eylem öneri sistemidir. Bu nedenle günümüzde, konum tabanlı sosyal ağlar ve konum tabanlı öneriler literatürde ilgi çeken araştırma konuları olarak ortaya çıkmışlardır.

Bireysel öneri yöntemlerindeki sorunlari gidermek amacıyla önerilmiş çeşitli melez sistemler bulunmaktadır. Bu tezde, genellikle iki boyutlu konum-eylem değerleme matrisi ile gösterilen coğrafi eylem öneri sisteminin doğruluğunu arttırmak amaçlanmaktadır. Bu iyileştirme, farklı kaynaklardan edinilen verilerin birleştirilmesi ve tekil değer ayrışımı (İng., SVD) yönteminin bilinen üstünlüklerinden yararlanılması ile gerçekleştirilmektedir. Bu yöntem, verinin içindeki gizli ilişkileri ortaya çıkarır ve büyüklüğünü azaltır. Ayrıca, tasarlanan yöntem, kullanıcı-konum-eylem formundaki üç boyutlu değerleme tensörünü iki boyutlu öneri matrisine çevirecek şekilde genişletilmiştir. Bu dönüştürme, yüksek seviye tekil değer ayrışımı (İng., HOSVD) ve çeşitli üç boyutlu matrislerin birleştirilmesi ile gerçekleştirilmiştir. Ek olarak, eylem-eylem benzerliği matrisi, anlamsal bir yapı içerisinden çıkarılmaktadır.

Anahtar sözcükler: Öneri Sistemleri, İşbirlikçi Filtreleme, Tekil Değer Ayrışımı, Özellik Kombinasyonu, Anlamsal Benzerlik, Yüksek Seviye Tekil Değer Ayrışımı

To my parents, who always supported and motivated me throughout my study.

ACKNOWLEDGEMENTS

I would like to appreciate Prof. Dr. İsmail Hakkı Toroslu for his encouragement and motivation throughout this work and his friendly support within all steps of preparing my thesis.

I honestly appreciate Assoc. Prof. Dr. Pınar Karagöz and also, Assoc. Prof. Dr. Murat Manguoğlu efforts. They honestly guided me throughout accomplishment of technical aspects of this work.

This work has been partially funded by the Greek GSRT (project number 10TUR/4-3-3) and the Turkish TUBITAK (project number 109E282) national agencies as part of Greek-Turkey 2011-2012 bilateral scientific cooperation.

TABLE OF CONTENTS

ABSTRACT v
ÖZ vi
ACKNOWLEDGEMENTS viii
TABLE OF CONTENTS ix
LIST OF TABLES xii
LIST OF FIGURES xiii
CHAPTERS
1 INTRODUCTION 1
1.1 Our Contribution 3
1.2 Organization of Thesis 3
2 RELATED WORK 5
2.1 Collaborative Filtering 5
2.1.1 Matrix Factorization 7
2.1.2 Probabilistic Models 7
2.2 Content based 8
2.3 Knowledge based 9
2.4 Hybrid Recommendation Systems 9
2.5 General Problems in Recommender Systems 11
2.6 Rating Issue 11
2.6.1 Explicit rating 11
2.6.2 Implicit rating 12
3 FEATURE COMBINATION MODEL AND IT'S USE IN RECOMMENDATION SYSTEM 13

3.1 Features and Their Extraction from Data 13
3.2 Singular Value Decomposition (SVD) Overview 14
3.3 Collaborative Matrix Factorization (CMF) 15
3.4 Integrated Feature Combination (IFC) 17
3.4.1 Feature Combination 17
3.4.2 Activity Recommendation 17
3.4.3 Running Example 18
4 DESCRIPTION OF EXTENDED FEATURE COMBINATION MODEL (EFC) 23
4.1 Problem Definition 24
4.1.1 User-User Similarity Matrix 25
4.1.2 Activity-Activity Similarity Matrix 26
4.1.3 Location-Location Similarity Matrix 26
4.2 Tensor Based Recommendation 27
4.2.1 Dimensionality Reduction 27
4.2.2 Recommendation Steps For Tensor-based Method 29
4.2.3 Example for Tensor Based Recommendation System 30
4.3 Extended Feature Combination (EFC) 32
4.3.1 Model Construction 32
4.3.2 Similarity Calculation 33
4.3.3 Rating Prediction 35
4.3.4 Feature Combination Using Collaborative filtering 37
4.3.5 Feature Combination Using Contextual Information 40
4.3.6 Hybrid Feature Combination 41
5 EXPERIMENTAL RESULTS AND EVALUATION 45
5.1 Data sets 45

5.1.1 CLAR Data set 45
5.1.2 Geosocial2 Data set 46
5.2 Prediction Evaluation Metrics 46
5.3 Recommendation Evaluation Metrics 47
5.4 Evaluation Method 48
5.5 Experiments for IFC 49
5.5.1 Parameter Tuning 51
5.6 Experiments for EFC 52
5.6.1 Selecting Neighborhood 53
5.6.2 Weight of Combination 53
5.6.3 Other Parameters 54
5.6.4 Prediction results 55
5.6.5 Recommendation results 56
6 DISCUSSION AND CONCLUSION 59
REFERENCES 61

LIST OF TABLES

Table 1. Methods of mixture in hybrid recommendation systems 10
Table 2. Predicted values of IFC vs. CLAR 21
Table 3. Sample location table 27
Table 4. Similarity matrices explanation 35
Table 5. Estimated values and parameters 37
Table 6. Comparison between models according to MAE, RMSE, and time of execution 56

LIST OF FIGURES

Figure 1. Proposed model for CLAR [45] 15
Figure 2. Objective function of CLAR model [45] 16
Figure 3. Gradients of variables [45] 16
Figure 4. Prediction using gradient descend [45] 16
Figure 5. Proposed integrated matrix model 25
Figure 6. Users' friendship network 25
Figure 7. Simple tree structure of activities 26
Figure 8. Unfolding of 3-order tensor [61] 28
Figure 9. Visualization of tensor reduction using HOSVD [61] 29
Figure 10. Sample rating tensor (T) 30
Figure 11. Components of tensor 30
Figure 12. Core tensor *C* 30
Figure 13. Similarity values of HOSVD 31
Figure 14. Similarity indices of HOSVD 31
Figure 15. Extended Feature Combination (EFC) models overview (empty sub-matrices contain zero values) 33
Figure 16. Reduced-rank visualization of *U* 34
Figure 17. Visualization of matrix trimming 34
Figure 18. Reduced-rank visualization of V^T 35
Figure 19. Collaborative filtering based combination model 38
Figure 20. Reduced rank matrices 38
Figure 21. Similarity indices of CF model 39
Figure 22. Similarity values of CF model 39
Figure 23. Similarity matrices 41
Figure 24. Additional extracted matrices 41
Figure 25. Indices of similarities 42

Figure 26. MAE values for proposed work vs. CLAR 49
Figure 27. RMSE values for proposed work vs. CLAR 50
Figure 28. MAE values for proposed work vs. CLAR with applying abstraction 50
Figure 29. RMSE values for proposed work vs. CLAR with applying abstraction 51
Figure 30. *RMSE* vs. parameter *m* without abstraction 51
Figure 31. *RMSE* vs. parameter *n* without abstraction 52
Figure 32. Neighborhood size vs. MAE 53
Figure 33. Percent of the data maintained in dimension reduction vs. MAE 55
Figure 34. Distance metrics of WordNet vs. MAE in Hybrid model 55
Figure 35. Impact of threshold on precision and recall (threshold=2) 57
Figure 36. Impact of threshold on precision and recall (threshold=3) 57
Figure 37. Impact of threshold on precision and recall (threshold=4) 57

CHAPTER 1

INTRODUCTION

From the beginning days of introduction to World Wide Web (W3) by Tim Berners-Lee [1] it has undergone fundamental changes in terms of content, scale, and users. At that time, data that was stored on the Web was very simple and light weight in sense of volume. It was not prevailing to be used by society and only few people especially scientist and computer professionals were among the common users of Web. Physical scale and distribution of Internet was not as huge and vast as it is at present time and it was available usually in scientific research centers and governmental supported companies.

As a result, finding a desired information or context was not a difficult task for users. But, with recent increase of Internet, geographical scale and technology of high speed communications channels from one side and profound increasing of users in number and level of familiarity with web technologies and tens of thousands gigabytes of data on the other hand, looking for a piece of data inside a bunch of registered websites can be a tough, complicated and time consuming task. Hence, we need a tool to present the data that best cover what users are looking for. As a solution at this time, information retrieval techniques and search engines have been introduced to digital world [2].Typical functionality of these systems can be presented as follows. When a user enters a term of interest to a search engine, it searches for matching content within entire data in Web according to its specific algorithm and consequently, the search engine ranks relevant pages regarding to its own optimized parameters and finally presents them to users.

One of the major sequences that caused by proliferation of electronic commerce and e-business services (buying products, product comparison, auction, electronic payment, etc.) is the growth of online stores and immense variety of items they offer. Major numbers of the users that visit these stores do not have sufficient personal experience to evaluate the huge number of items that the stores present. With a large number of options, there should be a solution that assists customers to find what they exactly looking for and avoid them from being overwhelmed by information overload [4].

A common belief to expand recommendation systems is the notion that, most of the times individuals trust on a recommendation about routine and daily activities made by friends or family [3]. For instance, a friend who enjoys watching science fiction movie and is familiar with your interest to this topic may recommend you to watch Star Wars. Either when you go

to a restaurant with a friend that has a cuisine similar to yours, you are eager to ask your friend to recommend a food from menu. Nevertheless, you may not like the food or book as much as your friend does or even you may dislike the item a friend recommended you. It is a usual feedback that is also applicable in real systems.

In agreement with mentioned belief, recommender systems (RS) are next generation of Web technology after information retrieval systems which are devised to propose an item to users according to their taste and tendency. Take into consideration that, we are talking about totally personalized recommendations, that is, every user sees a specific list of items depending on his or her personal taste. In fact, recommender systems are different from search engines regarding their fundamental functionality and objectives. We cannot generally include them in the same category as search engines and information retrieval systems. Burke in [41] mentions that it is the criteria of "individualized" and "interesting and useful" that separate the recommender system from information retrieval systems or search engines. The semantics of a search engine are "matching": the system is supposed to return all those items that match the query ranked by degree of match. Techniques such as relevance feedback enable a search engine to refine its representation of the user's query, and represent a simple form of recommendation.

Furthermore, to clarify this difference Jeffery M. O'brien [5] says: "The Web, they say, is leaving the era of search and entering one of discovery. What's the difference? Search is what you do when you're looking for something. Discovery is when something wonderful that you didn't know existed, or didn't know how to ask for, finds you."

In general, recommender systems are software tools and algorithms that intended to make suggestion of items to a user incorporating other users' behaviors and preferences, product descriptions and ratings that are given to visited items by users. Recommender systems are mainly utilized in commercial Web and e-commerce domain to attract more customers and provide them with products and links which best meet their preference and satisfaction to increase the number and diversity of items sold. These products, however, are not popular to all customers or included in top-selling items of shopping Websites [2].

The word "Item" is a general term which is usually used to demonstrate objects in various domains that systems recommend to users. This object ranges from a movie in Netflix[1] or MovieLens[2], a book in Amazon[3], a piece of music in Lasf.fm[4], a shopping list in eBay[5] or either an academic research paper in Scopus[6], to a social activity that can be done at a

[1] https://www.netflix.com

[2] http://www.movielens.org

[3] http://www.amazon.com

[4] http://www.last.fm

[5] http://www.www.ebay.com

[6] https://www.scopus.com

specific geographical location or even more interestingly, a friend request in social networks such as Facebook[7].

Nowadays, proliferation of wireless smart tool technologies such as smart phones, tablets, and Personal Digital Assistants (PDA) has influents modern life widely. In addition, most of these smart tools are equipped with Geographical Position System (GPS) and have an Internet connection via wireless networks or the connection that mobile operators provide with using SIM card services. In parallel, applications are developed to make use of these facilities. Location Based Social Network (LSBN) is an instance of the widely used applications that, user logs in to system and add his/her experience and comments about any activity that does in a specific location with its coordinate and time stamp. As a result, this information is gathered an organized as a meaningful form to recommend like-minded friends and activity to the user [23].

Activity recommendation is specially utilized by tourist or visitors that are not familiar with the town or area. When a user visits a new place and intend to do an activity, system may recommend him/her activities such as sightseeing, sports, and entertainment according to past preferences of current user and other users' activity history.

A recent work about this domain is done by Zheng et al. [46]. They have devised a system to track users' movement and save their trajectory. As an extended version of typical 2-D rating matrix, it is possible, in some domain, to present the rating within a 3-D matrix which is called *Tensor*. Symeonidis has done a complete work in [56] which argues the Tensor structure and different approaches to reduce its rank in order to utilize it in recommendation systems.

1.1 Our Contribution

In summary, we apply an integrated feature combination (IFC) method together with matrix factorization to enhance the performance of Geo-Activity recommendation systems. In addition, we extend IFC to define integrated feature combination (EFC) method to convert a 3-D tensor-based recommendation system to 2-D matrix-based system by means of generalizing the combination method and injection of semantically created activity similarity matrix. This extension improves the accuracy of recommendation in EFC comparing to tensor-based recommendation systems.

1.2 Organization of Thesis

In Chapter 2, we study the related work about recommendation systems. We also consider different recommendation techniques which have been proposed in literature. As well as their advantages and disadvantages are explained and possible approaches to deal with them are introduced.

[7] https://www.facebook.com

In Chapter 3, we introduce the integrated feature combination (IFC) method and the approach of utilizing it beside other classical RSs. A running example is added to describe the recommendation process and its calculation as well.

A brief introduction to the higher order singular value decomposition (HOSVD) and tensor-based recommendation technique is presented in Chapter 4. Additionally, extension feature combination (EFC) which is an extension to IFC is explained and corresponding running examples are shown as well.

Chapter 5 is dedicated to the evaluation part in which data sets and several evaluation metrics for estimation and prediction are explained. Meanwhile, various experiments for IFC and EFC have been performed and results are shown in this chapter.

And finally, we conclude the thesis in Chapter 6 with a brief discussion about the proposed method and obtained results for those methods.

CHAPTER 2

RELATED WORK

In this Chapter, we investigate major techniques of recommendation systems according to their basic strategy, addressed domain and knowledge used. We also study various aspects of each technique and possible advantage and disadvantage of them. Finally, we study common shortcomings of recommender systems and consider the latest solutions that have been proposed in the literature to challenge those problems.

2.1 Collaborative Filtering

The basic idea of collaborative filtering (CF) approach is to make use of information about the past behavior, opinions or feedback of available users about specific items to predict what item/s the current user of the system most probably interested in. Generally speaking, if users A and B had common interest (preference) for purchasing an item at past, it is likely that they will have similar tastes for other items at the future. Since selection of interesting items for a user requires filtering a lot of irrelevant items of similar-minded user and users inside the system collaborate with each other implicitly this method is known as *collaborative filtering* (CF) [2].

To find similarity between users or items, we should have a mechanism to store relation between items and users and additionally, collect feedback of users as well. The relation between users and items are modeled as a m by n matrix whose rows denotes users and columns denote items correspondingly. User feedback is kept as a *rating* value so that, if user u_i gives a rating about item i_j then this rating is inserted into the row i and column j of mentioned matrix. It should be noted that, for commercial systems, quantity of items and user can be a huge number and even exceeds more than 10,000 [25]. The matrix, consequently, is very sparse that means, there are several users that have not rated most of the items and their corresponding values within the matrix are not available. However, sparseness becomes a drawback for this technique. This issue is studied in [17] and [16], since it influences the performance of recommendation results.

In most of the applications, as an intrinsic property of stored information, recommendation problem can be reduced to an estimation problem to predict the value of unknown ratings for unrated items by specific users in mentioned matrix [20]. Zheng et al. in [45] proposes CLAR (Collaborative Location and Activity Recommendation) which models the missing

value prediction as Collective Matrix Factorization [51]. This method intends to predict the unknown rating by constructing a mathematical model. Then, using numerical methods such as Gradient Descent [49] tries to find maximum values of model. However, the drawback of Gradient Descent is that it does not promise to find absolute maxima and instead calculates local minima of model.

In general, algorithms for CF recommendation systems are divided into two categories [13] which are:

- Memory-based (or Heuristic-based)
- Model-based

Memory-based: These methods are essentially heuristics algorithms that stores whole user-item rating matrix and make predictions based on the entire collection of previously rated items by the users [17]. User-based nearest neighbor recommendation is a typical method of memory based algorithm in which, for a given active user similar k users, known as *neighbors*, using statistical methods are found in rating matrix. Afterwards, system considers the items that are rated with similar users but, not by active user, to calculate *top-N* recommendation for the active user [14].

However, metrics for finding similar users is an important issue and should be discussed in terms of performance. In the literature, several metrics such as *Pearson's correlation coefficient*, *cosine similarity*, *adjusted cosine similarity*, *Spearman*'s *rank correlation coefficient*, or the *mean squared difference* measure have been proposed to determine the similarity between users [75] which can influence performance of prediction.

A comprehensive work is conducted in [16] which evaluate the effect of each mentioned metric into recommendation result. In addition to metric determination, the value of k neighbors may affect final as well. According to [16], once the number of k neighbors is too high, there is a risk of "noise" and in contrast, too small k neighbors (below 10) misleads the predictions negatively.

Model-based: In this approach, instead of storing the user-item matrix system build a model from users' previous behaviors and utilizing probabilistic methods it can make a prediction of prospect item for an active user [14]. In other words, prior to make recommendation, raw data which is in form of rating matrix is preprocessed by system to train a learning model in offline mode. Then, this learned model is used to prepare rating prediction in run time.

The model building process is performed by different approaches such as *matrix factorization* or *probabilistic model*. Each of these approaches have its own method of implementation hence, several papers have been published in literature that we will consider some of them briefly.

2.1.1 Matrix Factorization

In this model, users and items can be inferred as row and column vectors. Consequently, rating matrix can be shown as product of distinct matrices that each one shows user space an item space. In information retrieval domain, respectively, it is called *latent semantic analysis* since it can remove noise and reveal some hidden correlation between user/item vectors [19].

One of the most widely used extensions of matrix factorization, *singular value decomposition* (SVD), proposed in [21] which can discover the latent factors in documents. Osmanli in [12] has analyzed the effect of different tag similarity techniques to the 3-Dimensional SVD recommendation performance. An application of SVD in geo-activity recommendation is argued in [18] as well. Besides, an in-depth study is done in [19] which, investigates advanced methods of matrix factorization for recommendation systems.

Nowadays, large CF-based recommendation systems in various domains support huge number of items and users. Additionally, regarding to consumer demands and information evolving it is getting even gigantic in terms of data dimensionality [9] and [25]. There are several algorithms that reduce this data dimensionality by mapping data from original size to lower dimensionality. Different techniques are proposed in literature including support vector decomposition [26], singular value decomposition [26], principal component analysis [8], and factor analysis [27].

2.1.2 Probabilistic Models

Another way of making a prediction about rating of user to a specific item is to utilize well-understood formalisms of probability theory [9]. Within the probabilistic models, Bayesian belief network is one of the popular techniques that, models the probabilistic dependencies among users or items. Breese et al. in [13] proposes a method to derive and apply Bayesian network using decision tree to demonstrate probability tables.

Further probabilistic approaches are argued in [28] and [29] which in latter work, some implementation detail of Google's news personalization engine is discussed. "Google News is a free news aggregator provided and operated by Google Inc., selecting most up-to-date information from thousands of publications by an automatic aggregation algorithm" [30]. Some research has exploited the advanced topics of probability models. For example, Shani et al. in [31] proposes Markov decision processes to model recommendation process.

Achieved results in [13] shows that, the comparison between probabilistic methods such as Bayesian networks outperform other techniques like user-based nearest neighborhood in some test domains. However, for movie recommendation domain, the Bayesian approach shows poor results comparing with some extended user-based techniques [17].

In essence, as discussed in [17] memory-based collaborative techniques calculate the desired ratings performing some heuristic computation across the entire database. They are simpler

and new data can be added easily and incrementally. However, when the size of user-item dataset grows, complexity of time and space increases as well. Whereas, model-based collaborative techniques, do not store whole data and create a statistical or probabilistic model based on available user-item data to predict the unknown ratings. Memory requirements for them are far less than memory-based. An amount of time is needed to analyze data and create a model from it and once a new item is added to system the mentioned process should be performed again.

An approach proposed by Pennock et al. in [32] which combines some of the advantages in memory-based and model-based CF together. According to empirical results that is done over movie ratings of EachMovie database and user profile data collected from the CiteSeer digital library of Computer Science research papers, they claim that this combination has a better performance than pure memory-based or model based recommendation systems.

In this thesis, we will make use of SVD's advantage in rank reduction which profoundly, alleviates the required time of recommendation since it purges the rating information from noise and according to achieved results it presents satisfactory performance.

2.2 Content based

So far, we have solely, we have focused only on the user-item rating matrix without either knowing the personal characteristic of user, or intrinsic properties of item to made recommendation. For instance, in Geo-activity recommendation system, we may recommend visiting Metropolitan Museum of Art for a user when he/she is in NY City, if we have a prior knowledge that the user is interested in Arts and Metropolitan Museum of Art is a place that usual activity performed there are related to Arts.

Content-based recommendation system is an approach that, besides using rating matrix, utilizes a description of the item characteristics and a user profile that somehow describes the (past) interests of a user. The recommendation task then consists of determining the items that match the user's preferences best [2]. Some technical details of building user profile are discussed in[33].

Text-based item (or text document) recommendation is one of fields that context-based recommendation system is used to recommend similar text-based item for a given keyword [17]. For instance, Balabanovic in [34] proposes a method to recommend Web pages to user using some informative keywords. Moony and Roy in [35] see the recommendation as a classification problem and make use of title, author, and reviews of a book and training a multi-nominal naïve Bayes classifier to recommend books for users. Other classification methods have been proposed in [36] which, they have exploited several machine learning algorithms such as decision trees and neural networks.

One of the limitations of content-based filtering is gathering attributes and features of users and items[17]. For users, mostly used method is to require the users to enter some personal information and fields of interests as well at the time of registration to the systems.

However, as discussed in [37] it is not so practical to insert the features of items manually. Instead, it is recommended to devise it so that, computer can extract them from content automatically. It is obvious automatic extraction for some kind of data type like image and video is a tricky task [17].

2.3 Knowledge based

In our daily life, we encounter some situations that, studied RSs are somehow unable to tackle them. To be more clear, buying a car, apartment, or a computer does not occur on a frequent basis. Since we do not provide recommendation systems with any feedback or preference about them, pure CF system will not perform well due to lack of sufficient number of ratings [38]. Furthermore, growing may change lifestyle of an individual which in turn causes variation on preferences. For example, recommending a fast food for a user, who used to eat hamburger but left this behavior and preferred to be a vegetarian does not make any sense and pure content-based system may not handle this situation.

Finally, in more complex product domains such as cars, customers often want to define their requirements explicitly – for example, "the maximum price of the car is x and the color should be black". The process of such demand is not manageable for pure collaborative and content-based recommendation systems [2].

Knowledge-based recommendation systems can assist us to deal with mentioned situations. As discussed in [38] and [39] "Recommenders that rely on knowledge sources not exploited by collaborative and content-based approaches are by default defined as knowledge-based recommenders" [2].

VITA is a knowledge-based financial service recommender application which was developed based on CWAdvisor method proposed by [40]. This system serves as sales representatives in sales dialogs with customers [40].

One of the advantages of knowledge-based systems is that they do not need to worry about cold-start problem. Since the knowledge is gathered in the system prior to creating recommendations. However, acquiring the needed knowledge is a tricky task that requires advanced data mining techniques to be accomplished.

2.4 Hybrid Recommendation Systems

As mentioned before, we focused on 3 categories of recommendation systems. We discussed advantage and disadvantage of each system and also studied some materials of literature about each technique. However, we need a system that can utilize the key points of each technique to increase the performance of recommendation. The rationale behind the hybrid recommendation system is to combine the strengths of different algorithms and exploit the synergy of all models to overcome some of shortcomings and problems that have studied in earlier research.

Burke in [38] defines the hybrid recommendation system as a combination of at least two typical recommendation system in order to improve recommendation performance.

According to recent work in [53], author has define the combination model of recommendation systems into seven categories that are showed in Table 1.

Table 1. Methods of mixture in hybrid recommendation systems

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.

A **feature combination** hybrid uses a diverse range of input data. For instance, Basu et al. in [42] proposed a feature combination hybrid that combines collaborative features, such as a user's likes and dislikes, with content features of catalog items. Another approach for feature combination was proposed by [43] which exploits different types of rating feedback based on their predictive accuracy and availability [2].

Hydra [52] is a movie recommender system that, utilize two distinct data sets to make recommendation. Practically, authors combine resources from *MovieLens*[8] rating data and *IMDB*[9] content information and then, they construct a mathematical model that recommends movie to users within a single system.

[8] http://www.grouplens.org/

[9] http://www.imdb.com/interfaces

Feature augmentation is another hybridization design that integrates several recommendation algorithms and then applies transformation steps on them. Content-boosted collaborative filtering is an example of this method [44]. It predicts a user's assumed rating based on a CF that includes content-based predictions.

2.5 General Problems in Recommender Systems

Depending on the recommendation system techniques which are used we may encounter various problems and difficulties in systems. These problems have been studied in literature [53] and we summarize some of them in this work.

Cold Start: As it is discussed in [53] cold start is a known problem that can be occurred in typical recommendation systems. In order to accomplish the task of recommending an item to a user, recommendation systems need to acquire some information about users and items in general words. However, when a new user register to a system for first time or a new item is added to the repository, we do not have any information about them. As a result, it is very unlikely that the recommendation system consider the new-coming user or item in early recommendations. Both collaborative filtering and content based filtering techniques suffer from the cold start problem. In order to alleviate the effect of cold start on final results, user give feedback to the systems by means of their ratings to items[17].

Data Sparsity: In a typical recommendation system we usually have a huge number of users and items. Most of the users have rated only very small portion of items and some items have received feedback from limited users. Therefore, in such a system rating values for majority of entries are unavailable. The work in [60] has studied the effect of data sparsity in collaborative filtering systems. The observed results show that the data sparsity influences the obtained results of collaborative filtering recommendation systems.

2.6 Rating Issue

As we have seen so far, most of recommendation techniques rely on user interaction with systems and their ratings to make a reliable recommendation. Hence, gathering user opinion or feedback about items is an essential issue which should be performed in a correct and quick form in recommendation systems. In general, this operation is divided to *explicit* rating and *implicit* rating that are discussed below [2].

2.6.1 Explicit rating

It is said that, the direct feedback of user represents correct opinion of him/her [2]. In many systems, this is done by asking directly from users about their experiences with bought commodities, watched movies or even visited places. This rating usually includes a five level scale in format of numbers or graphical signs such as stars, sliders and bars in which, the lowest number is inferred as "lowly interested" and the highest one as "highly interested" and the levels between them represent different stage of users' satisfaction.

However, optimal number of levels and understanding of users from them can be quite different in various domains. The effects of these issues are thoroughly argued in [7] and [8]. In addition, multi criteria rating recommender techniques are proposed which can be found in [6]. The problem with explicit rating is the time and effort that user should dedicate to interact with system. Sometimes, users are not eager to participate, since they find it boring and time-consuming.

2.6.2 Implicit rating

In this this method, users' opinion is collects as they log in to the system and begin to interact with system without recognizing that their activities are monitored. In practice, when a user purchases an item most of the RSs interpret this action as a "like" intension and may update rating for specific item [10]. For instance, when a user click on a video in YouTube, it deduce it as interest of the user about that content of video hence, a list of similar content videos are recommended for user to watch in sidebar.

Moreover, some systems may utilize the results of user's search, especially in online shopping Websites, and consider his/her spent time on specific item or page. According to defined period of time in system, RS can observe user tendency to specific item [2]. An obvious example of this method is found in Amazon so that, when a costumer searches for an item and add it to the shopping cart, even without going to the payment process, system interprets the item interested to user and recommends item/s similar to one in the user's shopping cart.

CHAPTER 3

FEATURE COMBINATION MODEL AND IT'S USE IN RECOMMENDATION SYSTEM

As we have briefly discussed in Chapter 2, using pure techniques of recommendation systems has several shortcomings. To compensate it, hybrid systems proposed which leverage recommendation task by means of combining available algorithms. *Feature combination* is one of the hybridization methods that acquires needed data from different sources and model them mathematically to use in a single RS. Furthermore, we studied model-based collaborative filtering which exploits ratings to predict missing values in user-item rating matrix.

In this Chapter, we briefly explain the features and the resource where they have been extracted. Then, present an overview to Singular Value Decomposition (SVD) method as a dimensionality reduction technique to uncover the latent relation within data. Afterwards, we proceed with description of the collaborative matrix factorization method and its recommendation procedure and finally, we give a detailed elaboration of integrated feature combination (IFC) model and the process of recommendation using that model.

3.1 Features and Their Extraction from Data

As we will explain later in this Section, we define features as additional data that usually are extracted from different sources of data. The data which is used to extract additional features is collected from a web-based application over 2.5 years so that, each user is equipped with a GPS installed tool (GPS Navigator, smart phone) during visiting Beijing city in China. For each location that is visited, users may insert comments about that place and available five activities that are done in that location. These five activities are food and drink, shopping, movie and shows, sports and exercise, tourism and amusement. Thus, these comments are processed and together with location information constitute Location-Activity matrix, whose rows are locations and columns are activities each entry shows the frequency of doing an activity. In the rest of this work we also will refer to this matrix with X.

To make it more informative, location-feature data extracted from *Point of Interest (POI)* database which is based on city's yellow pages, is also added to our model. Usually in big cities for any given location area in the city this database gives the type and the number of activities like cinemas, restaurants, shopping centers, sport complexes and so on. Gathered

data can be modeled as a Location-Feature matrix whose entries are nonnegative integer values that for a given location and will be shown by *Y*. Also, a 5-by-5 matrix exits in this data set which is named as Activity-Activity matrix whose entries are real values in interval [1,-1] and show the correlation between activities represented as the rows and columns. We will use the notation of *Z* to show Activity-Activity matrix.

3.2 Singular Value Decomposition (SVD) Overview

Singular Value Decomposition (SVD) [49] is a well-known matrix factorization technique that factorizes an $m \times n$ matrix *R* into three matrices as given in Equation (1):

$$R_{m\times n} = U_{m\times r}\ S_{r\times r}\ V^T_{r\times n} \tag{1}$$

Where, *U* (left singular vector) and *V* (right singular vector) are two orthogonal matrices of size $m \times r$ and $n \times r$ respectively; r is the rank of the matrix *R*. The eigenvectors of RR^T and R^TR make up the columns of *U* and *V* correspondingly. Also, values of *S* are called *singular values* and calculated from as roots of eigenvalues from RR^T or R^TR [57] [58].

In addition, matrix *S* is a diagonal matrix of size $r \times r$ having all singular values of matrix *R* as its diagonal entries. All the entries of matrix *S* are positive and stored in decreasing order of their magnitude. SVD provides the best lower rank approximations of the original matrix *R*, in terms of *Frobenius* norm [59].

We utilize SVD in recommender systems to perform two different tasks: First, we use it to capture latent relations among users and items that allow us to compute the predicted likeliness of a certain product by a customer. Second, we use SVD to produce a low-dimensional representation of the original user-item space and then compute neighborhood in the reduced space [57]. In general, SVD is used for dimensionality reduction so that, with selecting the k largest values of *S* and k columns of *U* and *V* (or k rows of V^T) and by multiplying them respectively, we can represent matrix *R* with reduced matrix R_k which has the rank of k ($k \leq Rank(R)$).

$$R_k = U_{m\times k}\ \ S_{k\times k}\ \ V^T_{k\times n} \tag{2}$$

However, dimensionality reduction is a data lossy approach. It means that, the more we reduce the dimensionality the big portion of original data is lost. It is possible to determine the percentage of loss in advance. According to study in [62] within image compression domain, to maintain p percent of original data we can select k singular values so that:

$$p = \frac{\sum_{i=1}^{k} S_k}{\sum_{all} S_k} \tag{3}$$

For instance, if singular values of matrix *M* are *S* ={10, 7.2, 5.4, 3.2, 3.1, 2.2, 1.9, 1.3, 0.8, 0.4, 0.1}; in order to keep 80% of original data, we should reduce the rank of matrix to 5 since:

$$\frac{\sum_{i=1}^{5} S_k}{\sum_{all} S_k} = \frac{28.9}{35.6} \sim 80\%$$

For simplicity, in the rest of the thesis we show the reduced rank components with U_k, S_k and V_k^T.

3.3 Collaborative Matrix Factorization (CMF)

In order to compare results of our proposed method, we have selected a cutting-edge research that is in domain of Geo-activity recommendation and discussed by Zheng et al. [45]. They have proposed a model-based based collaborative filtering technique based on *collective matrix factorization* [51] to predict missing values of target matrix which they name it *collaborative location and activity recommendation (CLAR)*. At the following, we will briefly explain their model.

Since the original Location-Activity is sparse and have many missing entries, authors propose to barrow some more information from location-feature and activity-activity matrices for prediction based on collaborative filtering. To do so, they try to build a model which combines location-feature and activity-activity matrices with original location-activity matrix. The model is shown in Figure 1.

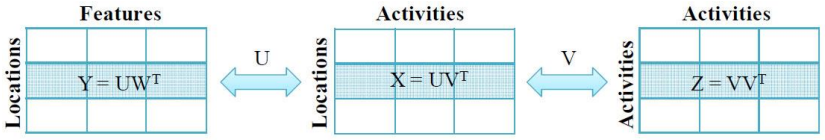

Figure 1. Proposed model for CLAR [45]

They decompose the location-activity matrix $X_{m\times n}$ by low-rank approximation as a product of two matrices $U_{m\times k}$ and $V_{n\times k}$ where $k < n$. It shares the location information through sharing matrix $U_{m\times k}$ with the location-feature matrix $Y_{m\times l}$, which is decomposed as a product of matrices $U_{m\times k}$ and $W_{l\times k}$. Similarly, the location-activity matrix shares the activity information through sharing matrix $V_{n\times k}$ with the activity-activity matrix $Z_{n\times n}$, which is decomposed as a self-product of $V_{n\times k}$. Then, utilizing a collective matrix factorization model they formulate an objective function which is depicted in Figure 2.

$$L(U,V,W) = \frac{1}{2} \| I \circ (X - UV^T) \|_F^2 + \frac{\lambda_1}{2} \| Y - UW^T \|_F^2 + \frac{\lambda_2}{2} \| Z - VV^T \|_F^2 + \frac{\lambda_3}{2} (\| U \|_F^2 + \| V \|_F^2 + \| W \|_F^2)$$

Figure 2. Objective function of CLAR model [45]

Where $\|.\|_F$ denotes the Frobenius norm. I is an indicator matrix with its entry $I_{ij} = 0$ if X_{ij} is missing, $I_{ij} = 1$ otherwise. The operator "$\circ$" denotes the entry-wise product which is called Hadamard product [63].

In this step, they try to minimize the objective function. However, it is discrete and regular algebraic methods cannot be applied to fine function's minimum. As a consequence, they intend to minimize the objective function, numerically, by assistance of gradient descent [50]. Generally speaking, this method is a type of hill climbing algorithm in artificial intelligence which attempts to find function's minimal values by stepping toward variations of gradient. Figure 3 shows gradient functions for each U, V, and W variables.

$$\nabla_U L = [I \circ (UV^T - X)]V + \lambda_1 (UW^T - Y)W + \lambda_3 U,$$
$$\nabla_V L = [I \circ (UV^T - X)]^T U + 2\lambda_2 (VV^T - Z)V + \lambda_3 V,$$
$$\nabla_W L = \lambda_1 (UW^T - Y)^T U + \lambda_3 W.$$

Figure 3. Gradients of variables [45]

Algorithm of gradient descent starts generally with a random value for function's parameters. Then within iteration it calculates gradients of function regarding to U, V, and W and updates current value by stepping toward direction that function has smallest slope. In the vicinity of minimum points, algorithm jumps out of loop probing a threshold variable in each step. As a result, missing values are predicted in and completed matrix can be used to make recommendation. Details of the algorithm are illustrated in Figure 4.

Algorithm CLAR

Input: Incomplete location-activity matrix $X_{m\times n}$, location-feature matrix $Y_{m\times l}$ and activity-activity matrix $Z_{n\times n}$.
Output: Complete location-activity matrix $X_{m\times n}$.

1. $t = 1$;
2. **While** ($t < T$ and $L_t - L_{t+1} > \epsilon$) **do** // T is #(max iterations)
3. Get the gradients ∇_{U_t}, ∇_{V_t} and ∇_{W_t} by Eq.(6);
4. $\gamma = 1$;
5. **While** ($L(U_t - \gamma\nabla_{U_t}, V_t - \gamma\nabla_{V_t}, W_t - \gamma\nabla_{w_t}) \geq L(U_t, V_t, W_t)$) **do**
6. $\gamma = \gamma/2$; // search for the maximal step size
7. $U_{t+1} = U_t - \gamma\nabla_{U_t}$, $V_{t+1} = V_t - \gamma\nabla_{V_t}$ and $W_{t+1} = W_t - \gamma\nabla_{W_t}$;
8. $t = t + 1$;
9. **Return** X;

Figure 4. Prediction using gradient descend [45]

3.4 Integrated Feature Combination (IFC)

Earlier we mentioned that, having a user-item rating matrix, some of the recommendation systems accomplish the recommendation task as prediction the unknown entries of rating matrix. That means, in domain of activity recommendation, if a user interested in doing a specific activity in a location and provide the system with his/her feedback as giving a rating then, in a new location that user visits system should predict the most probable ratings and present them as a list of activities to the user. In following sections we will discuss about our method for rating prediction.

3.4.1 Feature Combination

Since the Location-Activity is very sparse and this influences the recommendation performance of CF technique, we should devise a method to alleviate sparseness effect. On the other hand, we have additional data which is not explicitly informative about location or activity but, can be utilized indirectly to enhance system confidence. However, the major challenge is how we can inject additional data into a model together with Location-Activity matrix. We aim to reach this goal by means of feature combination.

Feature combination tries to integrate different data into a single matrix in order to enhance the performance of prediction. Consequently, we merge main matrix *X* (Location-Activity) with two additional matrices *Y* (Location-Feature) and *Z* (Activity-Activity) to construct an integrated matrix *T*. Equation (4) shows the combination in which we have merged three matrices in a single matrix *T* .Note that, in order to preserve the structure of a matrix we have injected zero values in the rest of the null entries of *T*. Subscripts in Equation (4) denote matrix dimension.

$$T_{(l+a)\times(f+a)} = \begin{bmatrix} Y_{l\times f} & X_{l\times a} \\ 0_{a\times f} & Z_{a\times a} \end{bmatrix} \tag{4}$$

So far, we have created the combination model successfully. In the following sub section, we will explain the procedure to predict unknown rating values in Location-Activity matrix (*X)*.

3.4.2 Activity Recommendation

Decomposing original matrix *T* reveals an interesting characteristic of U_k and V_k^T. Actually, using $U_k$for a given activity we can easily find similar locations that this activity is also done and using V_k^T, for a given location we can find similar activities that are done in that location. Moreover, combining these two may even lead to better results. In this step, to predict a frequency rating for a given location *i* and activity *j*, $a_{i,j}$we search for similar rows of *i* in U_k and also for similar columns of j in V_k^T. In order to have an accurate estimate for frequency rating, instead of selecting one similar neighborhood, we pick the most similar *m*

rows in U_k and also the most similar *n* columns in V_k^T. Equations (5) and (6) define these operations.

$$M\_Rating_{sim\ row} = Mean(\sum_{s} Rating(a_{s,j})) \quad s = top\ m\ similar\ rows\ in\ U_k \tag{5}$$

$$M\_Rating_{sim\ col} = Mean(\sum_{s} Rating(a_{i,s})) \quad s = top\ n\ similar\ columns\ in\ V_k^T \tag{6}$$

After those operations, the average of $M\_Rating_{sim\ row}$ and $M\_Rating_{sim\ col}$ is used as the predicted rating value for performing activity a in location l.

Notice that, since the row count of U_k is more than the number of actual locations (due to merge operation) to find similar locations in U_k we trim it so that, its rows corresponds to locations (*l*) only. This is done by selecting the first *l* rows of U_k for the similarity search. Similarly, to find similar activities in V_k^T we trim it so that, its columns corresponds to the activities (*a*) only. It is performed by selecting last *a* columns of V_k^T for the similarity search. Thus, equations (4) and (5) are applied to only these rows and columns.

In order to see influence of similarity metrics, we have applied both Euclidean distance and Cosine similarity to get similarity matrices. On the basis of our experiments, we have observed that, Cosine similarity yields more accurate results than Euclidean distance. Cosine similarity is given in Equation (7) where, $\|\boldsymbol{x}\|$ denotes the Euclidean norm of $\boldsymbol{x}$.

$$sim(\boldsymbol{a}, \boldsymbol{b}) = \frac{\boldsymbol{a}^T \boldsymbol{b}}{\|\boldsymbol{a}\|\ \|\boldsymbol{b}\|} \tag{7}$$

3.4.3 Running Example

In this section we demonstrate a sample example of our method with a simplified data set. As we have already seen, $X_{5\times3}$ is Location-Activity matrix, $Y_{5\times4}$ is Location-Feature matrix, and $Z_{3\times3}$ is Activity-Activity correlation matrix.

$$X = \begin{bmatrix} 1 & 3 & 1 \\ 0 & 17 & 17 \\ 0 & 53 & 53 \\ 76 & 4 & 82 \\ 2 & 0 & 0 \end{bmatrix} \tag{8}$$

$$Y = \begin{bmatrix} 0.0037 & 0.0037 & 0.0038 & 0 \\ 0 & 0 & 0 & 0 \\ 0.0082 & 0.0082 & 0.0165 & 0.0131 \\ 0.0100 & 0 & 0.0100 & 0.0318 \\ 0 & 0.0757 & 0 & 0 \end{bmatrix} \tag{9}$$

$$Z = \begin{bmatrix} 1 & 0.0650 & 0.0008 \\ 0.0650 & 1 & 0.0017 \\ 0.0008 & 0.0017 & 1 \end{bmatrix} \quad (10)$$

At the beginning, we combine *X*, *Y* and *Z* matrices based on (1) to construct the integrated matrix *T*. In order to illustrate how our method works and how accurately it determines some missing values, we select 3 nonzero entries from *X* randomly and change their values to 0. The selected entries are x_{11}, x_{22} and x_{33} which are bolded in *T*.

$$T = \begin{bmatrix} 0.0037 & 0.0037 & 0.0038 & 0 & \mathbf{0} & 3 & 1 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{0} & 17 \\ 0.0082 & 0.0082 & 0.0165 & 0.0131 & 0 & 53 & \mathbf{0} \\ 0.0100 & 0 & 0.0100 & 0.0318 & 76 & 4 & 82 \\ 0 & 0.0757 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0.0650 & 0.0008 \\ 0 & 0 & 0 & 0 & 0.0650 & 1 & 0.0017 \\ 0 & 0 & 0 & 0 & 0.0008 & 0.0017 & 1 \end{bmatrix}$$

According to [48], SVD method is applied to *T* which yields matrices $U_{8\times8}$, $S_{8\times7}$ and $V^T_{7\times7}$ as follows:

$$U = \begin{bmatrix} \mathbf{-0.025} & \mathbf{-0.010} & -0.999 & 0.035 & 0.001 & -0.016 & 0.005 & 0 \\ \mathbf{-0.279} & \mathbf{-0.124} & 0.008 & -0.031 & 0.952 & -0.008 & 0.008 & 0 \\ \mathbf{-0.870} & \mathbf{-0.385} & 0.026 & 0.010 & -0.305 & 0.003 & -0.003 & 0 \\ \mathbf{-0.405} & \mathbf{0.914} & 0 & -0.026 & -0.001 & 0 & 0.001 & 0 \\ \mathbf{-0.010} & \mathbf{0.024} & 0.035 & 0.999 & 0.033 & 0.001 & 0 & 0 \\ -0.001 & 0.001 & 0 & 0 & 0.006 & -0.302 & -0.953 & -0.001 \\ 0 & 0 & -0.017 & 0 & 0.010 & 0.953 & -0.302 & 0.004 \\ 0 & 0 & 0 & 0 & 0 & -0.004 & 0 & 1 \end{bmatrix}$$

$$S = \begin{bmatrix} \mathbf{79.36} & \mathbf{0} & 0 & 0 & 0 & 0 & 0 \\ \mathbf{0} & \mathbf{75.48} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2.12 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0076 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.00069 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0000058 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.0000017 \end{bmatrix}$$

$$V^T = \begin{bmatrix} -0.00001 & 0.00001 & -0.0002 & -0.001 & \mathbf{-0.36} & \mathbf{-0.63} & \mathbf{0.69} \\ -0.00001 & -0.000002 & -0.00004 & 1 & \mathbf{-0.01} & \mathbf{0.01} & \mathbf{0.004} \\ -0.00002 & 0.000004 & -0.0002 & 0.0005 & -0.73 & -0.27 & -0.63 \\ -0.00003 & 0.00003 & 0.00002 & -0.009 & -0.58 & 0.73 & 0.36 \\ -0.39 & 0.92 & 0.04 & -0.0000002 & 0.00003 & -0.00002 & -0.00002 \\ -0.66 & -0.25 & -0.71 & -0.00004 & 0.0001 & 0.0001 & -0.000008 \\ -0.64 & -0.30 & 0.71 & 0.00002 & -0.0001 & -0.0001 & 0.00001 \end{bmatrix}$$

Since we are interested in specific part of *U* and V^T, we trim them so that, they meet the same indices of *X* in *T*. In order to reduce the integrated matrix to rank 2, first 2 columns of *U*, *S* and first 2 rows of $V^T$are selected as shown in Equations (11), (12), and (13) respectively.

$$U_k = \begin{bmatrix} -0.025 & -0.01 \\ -0.279 & -0.124 \\ -0.87 & -0.385 \\ -0.405 & 0.914 \\ -0.01 & 0.024 \end{bmatrix} \quad (11)$$

$$S_k = \begin{bmatrix} 79.36 & 0 \\ 0 & 75.48 \end{bmatrix} \tag{12}$$

$$V_k^T = \begin{bmatrix} -0.36 & -0.63 & 0.69 \\ -0.01 & 0.01 & 0.004 \end{bmatrix} \tag{13}$$

To predict the value of x_{ij} in *X* we search for similar rows of *i* in U_k and similar columns of *j* in V_k^T. Remember that to compute similarity matrix, we made use of Cosine similarity between vectors. Related similarity matrices are presented in Equations (14) and (15).

$$Sim(U) = \begin{bmatrix} 1 & 1 & 0.9352 & 0.6974 & 0.9997 \\ 1 & 1 & 0.9356 & 0.6983 & 0.9997 \\ 0.9352 & 0.9356 & 1 & 0.906 & 0.9441 \\ 0.6974 & 0.6983 & 0.906 & 1 & 0.7158 \\ 0.9997 & 0.9997 & 0.9441 & 0.7158 & 1 \end{bmatrix} \tag{14}$$

$$Sim(V^T) = \begin{bmatrix} 1 & 0.998 & 0.511 \\ 0.998 & 1 & 0.461 \\ 0.511 & 0.461 & 1 \end{bmatrix} \tag{15}$$

Using these similarity matrices we compute *Sim_index(U)* and *Sim_index(V^T)*. Each row in *Sim_index (U)* shows the index of most similar rows in descending order from left to right. Similarly, each column in *Sim_index(V^T)* shows the index of most similar columns in descending order from top to down.

$$Sim\_index(U) = \begin{bmatrix} 2 & 5 & 3 & 4 & 1 \\ 1 & 5 & 3 & 4 & 2 \\ 5 & 2 & 1 & 4 & 3 \\ 3 & 5 & 2 & 1 & 4 \\ 2 & 1 & 3 & 4 & 5 \end{bmatrix} \tag{16}$$

$$Sim\_index(V^T) = \begin{bmatrix} 2 & 1 & 1 \\ 3 & 3 & 2 \\ 1 & 2 & 3 \end{bmatrix} \tag{17}$$

As a final step, we are to predict the value of given x_{ij} from Location-Activity matrix *X*. We find the mean value of top *m* (in this example *m* is chosen to 3) similar nonzero rows of *i* and mean value of top *n* (in this example *n* is chosen to 1) similar nonzero columns of *j* incorporating Equations (16) and (17). The estimated value of x_{ij} is mean value of these 2 terms. Prediction steps for x_{11}, x_{22} and x_{33} are as follow.

Top 3 similar rows of row 1 are 2, 5 and 3 with corresponding values of 0, 2 and 0 in column 1. Since values of rows 2 and 3 are zero we put them aside and select next similar rows which in this case is row 4 only.

$$Mean\ (1,76) = 38.5$$

According to first column of Equation (17), column 2 is the most similar column to column 1 with value of 3 in row 1. Thus, the predicted value for x_{11} is mean value of this value and the value that is calculated as:

$$Mean\ (38.5, 3) = 19.25$$

In x_{22}, top 3 nonzero similar rows of row 2 are 1, 3 and 4 with values of 3, 53 and 4 in column 2.

$$Mean\ (3, 53, 4) = 20$$

Also, column3 is the most similar nonzero column of column 2 with value of 17 in row 2.

$$Mean\ (20, 17) = 18.5$$

Procedure for x_{33} is same as the previous entries thus, we just show the values.

$$Mean\ (17, 1, 82) = 33.33$$

$$Mean\ (33.33, 53) = 43.16$$

In order to have a better comparison, we have shown the predicted values of IFC and CLAR with original values at Table 2.

Table 2. Predicted values of IFC vs. CLAR

	IFC	**CLAR**	**Original**
x_{11}	19.25	10.4205	1
x_{22}	18.5	15.3352	17
x_{33}	43.16	7.6185	53

CHAPTER 4

DESCRIPTION OF EXTENDED FEATURE COMBINATION MODEL (EFC)

In a typical collaborative filtering recommendation model the input is the rating matrix corresponding to users and items. However, in many real world data there are additional features that could be useful. Mainly, the following extensions are possible:

- In recent applications, the rating data includes additional dimensions to user and item in the form of context information mostly involving time and location. Typically the context is time or place, or both. This addition changes the main data structure of the problem from 2-D to 3-D (or even higher). Some recent studies on this direction are [65][80][81].

- In some applications there are additional data about the objects of the dimensions that could be useful. Usually this data represent the relationships, more specifically similarities, among the objects of the dimensions. Indeed when there is no rating data, and only these similarities are used that could be interpreted as a form of content based filtering. Recent studies on extending rating data with additional features for standard 2-D rating matrix can be found in [18][46].

- There are also some applications (hybrid models) with both multi-dimensional rating data and additional features' data on the objects of the dimensions [46][79].

The main objective of this Chapter is to develop a uniform model suitable for all the models explained above. To achieve this multidimensional rating data (we will focus on 3-D form) will be reduced to 2-D form which can potentially be extended with the additional features.

The extension of 2-D rating matrix with additional features has been introduced in [18]. It has also been shown that the approach is quite effective and outperforms other more complicated solutions, such as [46]. There are works that specifically attempt to deal with 3-D rating data input [61][65][80]. In this work we show that 3-D rating matrix can be reduced to three 2-D matrices with a very little information loss. Our evaluations show that our reduction based model outperforms 3-D based model inspired from [61].

Some methods use ratings that are given by users to find similar objects (Generally users and items) within the system to fulfill the recommendation [69][70]. Some others use the

context-based information about users and items to catch a probable similarity between them which is used to find like-mind users and similar items[79]. Furthermore, some methods try to combine several sources of data with different concepts to improve the precision of the final results [81][52]. The latter one is indeed the approach we have extended in our previous work in [18] and in this paper we aim to extend the given approach.

4.1 Problem Definition

In recommender systems' domain, conventionally 'user – item' matrix is used to denote and keep the rating given by a user for an item. However, considering the characteristic of the data and the field which recommender system is used, these objects and even their dimensions may change. Social activity recommendation is one of the domains in which the types of objects are changed and even another dimension is added to the original data. That is, we have user, location, and activity as objects and high dimensional data structure (tensor) is used to handle the existent ternary relation among them. In social activity recommendation, beside the original user-location-activity rating matrix, we might have access to different informative resources of data. As we have already explained in [18], it is possible to acquire several data and merge them into a single integrated matrix and propagate the effect of additional data to the main part of the matrix. Observed results for this model show that the matrix merging can influence the process of recommendation and actually, improves the final results.

In this work, we aim to extend the model given in [18] by exploiting even more data and merge them to create a single matrix, which further is utilized as an input to the recommendation model. One of the problems in creation of integrated matrix is to maintain its shape. That means we inject additional data so that, in addition to make use of its informative content, we should keep the structure of final matrix in a rectangular shape. In order to maintain its structure we might need to insert zero value to the portions that its corresponding data is not available. Sometimes we merge a matrix into the final model such that in other portion, transpose of the matrix might also be usable for another portion of final model. For example, if location-activity is inserted to the final model, it is obvious that its transpose activity-location might be also inserted to it. However, this situation may have negative influences since it can make the model biased towards location-activity matrix. In order to prevent this effect, we only use one form of a matrix and if necessary, we insert zero instead of other form (either normal or transpose) of it.

In order to get activity-user, location-user, and location-activity we should aggregate tensor T over location, activity, and user dimensions respectively. For instance, to obtain location-activity matrix we aggregate the original tensor over user's dimension. In most of the rating data, each entry shows a rating value between 1 and 5. After summing up rating values of user's dimension, to calculate the value of an entry in location-activity matrix, we find mean value of ratings for all users who have rated activity a in location l.

In this new model, we propose to exploit location-location, activity-activity, and user-user similarity matrices in addition to activity-user, location-activity, and location-user data. The visualization of the model is shown in Figure 5.

	Activity	**Location**	**User**
Activity	Activity Activity ***A***		
Location	Location Activity ***B***	Location Location ***C***	
User	User Activity ***D***	User Location ***E***	User User ***F***

Figure 5. Proposed integrated matrix model

4.1.1 User-User Similarity Matrix

We construct User-User similarity matrix by using the friendship network among the users. From several distance metrics that are available such as *Graph Distance*, *Common Neighbors*, *Jaccard's Coefficient*, *Adamic & Adar* and *Preferential Attachment* [75], we choose *Jaccard's Coefficient*, which is well suited for our data set. Its definition is given in Equation (18).

$$Jaccard's\ Coefficient(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|} \tag{18}$$

In this definition, $\Gamma(x)$ and $\Gamma(y)$ are first level neighbours of x and y. Figure 6 shows a small sample of friendship network out of 149 users' friendship graph.

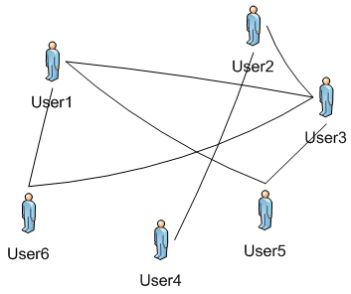

Figure 6. Users' friendship network

In order to find the similarity between *user 1* and *user 2* in this friendship network, we first find proximity of each user and then calculate *Jaccard's* similarity as given below calculations:

$$\Gamma(u_1) = \{u_3, u_5, u_6\} \qquad \Gamma(u_2) = \{u_3, u_4\} \qquad similarity(u_1, u_2) = \frac{|\{u_3,u_5,u_6\} \cap \{u_3,u_4\}|}{|\{u_3,u_5,u_6\} \cup \{u_3,u_4\}|} = 0.25$$

4.1.2 Activity-Activity Similarity Matrix

As also stated in various resources in the literature[45], various activities that we do in normal life are not independent and have a relation. A simple method to find the correlation between activities is proposed in [46], such that for any two activities, web is searched and from retrieved total and related websites, similarity between these activities is calculated. Another technique, which may help us to find this relation, is using activity ontology. In this model, each activity is denoted as a vertex and a relation between two vertices is shown by an edge. However, the conceptual structure may be a simple tree structure as shown in Figure 7 or may be obtained from available ontologies in the the web. *WordNet* is a large lexical database of English nouns, verbs, adjectives, and adverbs which labels the semantic relations among words [74]. In order to find similarity between words over WordNet several distance metrics proposed in the literature [82][83][84]. In this work, we will utilize those metrics to find similarity between activities [85][86].

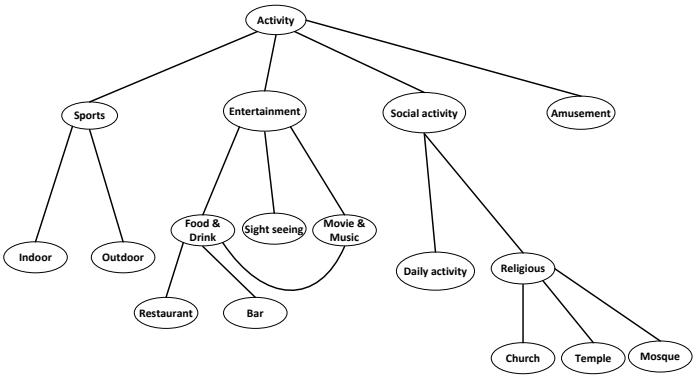

Figure 7. Simple tree structure of activities

4.1.3 Location-Location Similarity Matrix

In social activity recommendation systems and in location based social networks, coordinates of visited locations are kept in terms of geographical altitude and longitude. Further information such as location address, district, and city may be available as well. In order to find the similarity between two locations, it is sufficient to use simple Euclidian distance calculation. Table 3 shows sample location information.

Table 3. Sample location table

Location Name	X coordinate	Y coordinate	Address	District	City
Starbucks	22.95	40.63	Egnatias 123	City Center	2
Goodys	22.93	40.63	Dodekanisou 7	Vardaris	2
McDonalds	23.73	37.97	Ermou 2	Sintagma Square	1
Hard Rock Cafe	23.73	37.97	Filellinwn 18	-	1

4.2 Tensor Based Recommendation

A *Tensor* is a multi-dimensional matrix which is used as rating storage in recommendation systems. Specifically, it is used in Geo-activity, in which there exist relations among three entities of User, Location, and Activity. As discussed earlier, users in such a system can submit their feedback as a rating for any activity that they perform in geographically different locations. As a result, we can define 3-order (3-D) tensor so that, the dimensions represent User, Location, and Activity, respectively. Each entry of form (i, j, k) is a numerical value which shows the rating that is given by user i for performing activity k at location j.

Moreover, the high-order singular value decomposition (HOSVD) [64] is generalized to apply SVD to tensors. Cutting-edge researches are conducted by Symeonidis et.al. [61] and Marinho et.al. [68] which propose a unified social tagging recommendation system model exploiting HOSVD. We briefly explain their model later on in this section and extend our proposed model to convert a tensor-based recommendation to an integrated matrix-based model.

4.2.1 Dimensionality Reduction

Initially, the system in [61] begins the tensor reduction process by giving matrix-shape representation of rating tensor $\mathcal{A}$. Applying unfolding to $\mathcal{A}$ yields three matrices A_1, A_2 and A_1which are defined as follows:

$$A_1 \in R^{I_l \times I_u I_a}, \qquad A_2 \in R^{I_u \times I_l I_a}, \qquad A_3 \in R^{I_l I_u \times I_a}$$

Three unfolding matrices which are called 1-mode, 2-mode, and 3-mode are visualized in Figure 8. In the next step, regular SVD is applied to each unfolding matrix which results with total 9 new matrices.

$$A_1 = U^{(1)} . S_1 . V_1^T, \qquad A_2 = U^{(2)} . S_2 . V_2^T, \qquad A_3 = U^{(3)} . S_3 . V_3^T \tag{19}$$

Despite regular SVD in which we select *k* singular values to reduce the rank of matrix, for each mode of tensor we should determine three distinct parameters k_1, k_2, and k_3 since SVD is applied to A_1, A_2, and A_3 independently. The parameters k_1, k_2, and k_3 are empirically chosen by preserving a percentage of information of the original S_1, S_2, S_3 matrices after appropriate tuning [61]. As we discussed in previous sections, we can determine the percentage that we desire to maintain from original data. In [61] Symeonidis et.al. mentions that, the percentage is usually set to be *50%* of each unfolded matrix.

Since they have determined the rank of left singular vectors for all three unfolding, they proceed to construct the core tensor *S*. Core tensor acts as an intermediate which can control the interaction between user, location , and activity. Core tensor *S* is constructed as shown in Equation (20).

$$S = \mathcal{A} \times_1 U_{k1}^{(1)^T} \times_2 U_{k2}^{(2)^T} \times_3 U_{k3}^{(3)^T} \tag{20}$$

In this equation, $\mathcal{A}$ is the original tensor, $U_{k1}^{(1)}$, $U_{k2}^{(2)}$ and $U_{k3}^{(3)}$ are reduced matrices of right singular vectors by rank of k_1, k_2, and k_3 respectively.

In addition, $\times_n$ is defined as the *n*-mode product of an *N*-order tensor $\mathcal{A} \in R^{I1 \times I2 \times \ldots \times IN}$ by a matrix $U \in R^{Jn \times In}$. The result of product is an $I_1 \times I_2 \times \ldots \times I_{n-1} \times J_n \times J_{n+1} \times \ldots \times I_N$-tensor whose entries are defined as presented in Equation (21).

$$(A \times_n U)_{i_1 i_2 \ldots 1_{n-1} j_n i_{n+1} \ldots i_N} = \sum_{i_n} a_{i_1 i_2 \ldots i_{n-1} i_n i_{n+1} \ldots i_N} u_{j_n i_n} \tag{21}$$

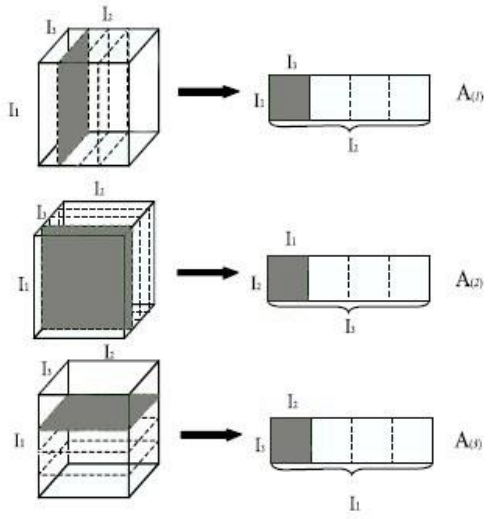

Figure 8. Unfolding of 3-order tensor [61]

Finally, tensor $\hat{\mathcal{A}}$ is constructed from product of core tensor *S* and dimensionally reduced left singular vectors of each unfolding modes as demonstrated in Equation (22).

$$\hat{\mathcal{A}} = S \times_1 U_{k1}^{(1)} \times_2 U_{k2}^{(2)} \times_3 U_{k3}^{(3)} \tag{22}$$

As discussed in [61], authors expect to extract latent semantic relation from $\hat{\mathcal{A}}$ as well as the value of missing entries in $\mathcal{A}$. A visual description of HOSVD is depicted in Figure 9.

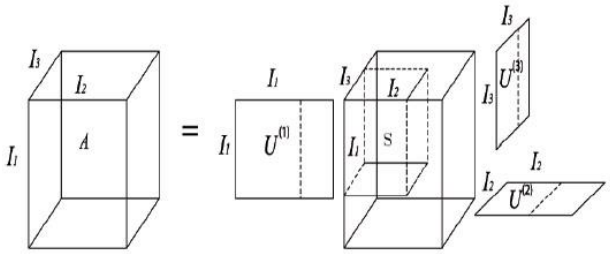

Figure 9. Visualization of tensor reduction using HOSVD [61]

So far, we briefly outlined the approach of HOSVD to reduce the rank of a given tensor $\mathcal{A}$. In the next subsection, we will describe the recommendation method exploiting rank-reduced tensor $\hat{\mathcal{A}}$.

4.2.2 Recommendation Steps For Tensor-based Method

Tensor $\hat{\mathcal{A}}$ presents the associations among the entities and serves as rating data model that is used during the recommendation. In order to predict the rating value that is given by user i for performing activity k in location j, firstly, we find most similar neighbors of user i from reduced tensor. Since each of $U_{k1}^{(1)}$, $U_{k2}^{(2)}$ and $U_{k3}^{(3)}$ corresponds to user, location, and activity respectively, we employ $U_{k1}^{(1)}$ to find similar users. Cosine similarity is used as the metric to accomplish this task. Secondly, we freeze the indices of location and activity and retrieve the rating values given by similar neighbors of user i for doing activity k in location j. Finally, we combine the rating values of each neighbor utilizing Equation (35) to produce a partial prediction from user. The routine of freezing indices and c producing similar neighborhood is applied to $U_{k2}^{(2)}$ and $U_{k3}^{(3)}$ to calculate location similarity and activity similarity as well.

As discussed in the previous subsection, applying HOSVD to tensor $\mathcal{A}$ yields three matrices $U^{(1)}$, $U^{(2)}$ and $U^{(3)}$. Considering the relation among entries, $U^{(1)}$ is the left singular vectors matrix of A_1 that is the result of aggregation on original tensor over location and activity. In other words, each row of $U^{(1)}$ is a vector for observation of user having location and activity together as the second dimension. This fact also holds for $U^{(2)}$ and $U^{(3)}$ where the former one is observations of location regarding to user and activity and the latter one is observations of activities regarding to user and location. This fact allows us to use these three matrices as a source to find similarity among entries. For example

choosing $U^{(1)}$ and any similarity metric (Euclidean, Cosine…), we can construct a similarity matrix of size user-count × user-count.

4.2.3 Example for Tensor Based Recommendation System

As a reduced scale example, suppose we have a rating tensor of 5 users in 5 locations for 4 activities. We are going to predict the values of random entries $a_1 = (2,1,2)$, $a_2 = (5,2,2)$ and $a_3 = (5,1,4)$ which are underlined in Figure 10.

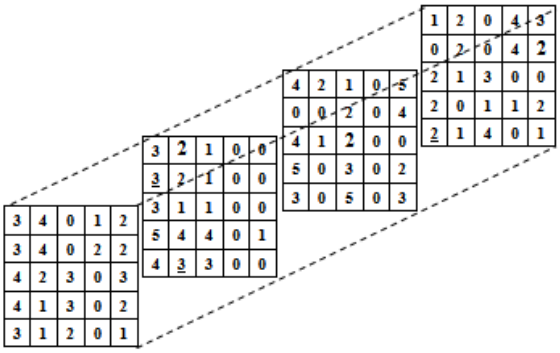

Figure 10. Sample rating tensor (T)

As discussed in before, prior to starting the procedure in order to illustrate how the model works, we replace the values of randomly selected entries (a_1, a_2, a_3) with zero. The process begins by producing unfolding modes then SVD is applied to each mode that results in matrices that are shown in Equation (19). Applying HOSVD to this tensor, according to Figure 9 and reducing to rank 2, yields three matrices U_1, U_2, U_3 and a core tensor *C* which are shown in Figure 11 and Figure 12.

0.4626	0.5746
0.3841	0.5481
0.3682	-0.3362
0.531	-0.3092
0.47	-0.4008

a) User (U_1)

0.7008	-0.1558
0.3894	0.3569
0.4561	-0.5564
0.0923	0.5754
0.3751	0.4556

b) Location (U_2)

0.5558	0.1347
0.4808	0.6853
0.5743	-0.2942
0.3606	-0.6525
0.5558	0.1347

c) Activity (U_3)

Figure 11. Components of tensor

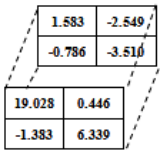

Figure 12. Core tensor *C*

In order to find similarity between entries, as discussed before, Cosine similarity is used as the distance metric. The similarity matrices and related indices are shown in Figure 13 and Figure 14. Note that, similarity between each object and itself is equal to 1 and in order to show a correct relation among similar objects, values of diagonals in similarity matrices are set to 0.

0.9978	0.9978	0.9994	0.984	0.9994
0.1499	0.0839	0.9775	0.9775	0.984
0	0	0	0.1499	0
-0.0284	-0.0948	-0.0622	0.0839	-0.0284
-0.0622	-0.1284	-0.1284	0	-0.0948

a) Sorted value of similarity from U_1

0.7867	0.9902	0.7867	0.863	0.9902
0.573	0.7839	0	0.7839	0.863
0.4528	0.573	-0.0552	0	0.4528
0	0	-0.1943	-0.0598	0
-0.059	-0.055	-0.6633	-0.6633	-0.194

b) Sorted value of similarity from U_2

0.7576	0.7509	0.8296	0.8296
0.7509	0.1379	0.7576	0.264
0.264	0	0.1379	0
0	-0.4386	0	-0.4386

c) Sorted value of similarity from U_3

Figure 13. Similarity values of HOSVD

2	1	5	5	3
4	4	4	3	4
1	2	3	1	5
5	5	1	2	1
3	3	2	4	2

a) Sorted index of similarity from U_1

3	5	1	5	2
2	4	3	2	4
5	1	2	4	1
1	2	5	1	5
4	3	4	3	3

b) Sorted index of similarity from U_2

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

c) Sorted index of similarity from U_3

Figure 14. Similarity indices of HOSVD

In the final step, for randomly selected entry $a_1 = (2,1,2)$, we keep values of $location = 1$ and $activity = 2$ then we find the most similar users (the size of neighborhood in this sample is 1) to user 2 from Figure 13.a which is user 1. The rating value for $T(1,1,2)$ is 3 hence first prediction can be calculated as:

$$P_{User} = 2.5833 + \frac{(3 - 2.5333) * 0.9978}{|0.9978|} = 3.05$$

Then, we keep values of $user = 2$ and $activity = 2$ and find the most similar locations to location 1 using Figure 13.b. According to activity-activity matrix that similar locations is 3.

$$P_{Location} = 3.2222 + \frac{(1 - 2.4375) * 0.7867}{|0.7867|} = 1.79$$

Finally, we freeze values of $user = 2$ and $location = 1$ then find the most similar activity to activity 2. Regarding to Figure 13.c it is activity 1 hence:

$$P_{Activity} = 2.5625 + \frac{(3 - 2.5) * 0.7509}{|0.7509|} = 3.06$$

At the end, we define the final prediction for $a_1 = (2,1,2)$ as the mean value of three initial predictions which is:

$$\hat{T}(2,1,2) = \frac{[1\ 1\ 1].[3.05\ 1.79\ 3.06]^T}{3} = 2.63$$

4.3 Extended Feature Combination (EFC)

When SVD is applied to a single user-item matrix, it produces left singular vectors *U* and right singular vectors *V* matrices. Each row of *U* represents a user object and each row of V represents an item object which both of them together with *S* (Singular Values) can be used to reduce the dimension of original matrix and finding the similarity within users and items as well. In this work we intend to make use of additional data and propagate their influences to other data matrices in order to improve the accuracy of recommendation.

Generally speaking, the basic step in most of recommendation systems begins by predicting unknown values in rating matrix. Afterwards, item recommendation is performed according to some threshold of estimated rating. Earlier we mentioned that, for a given user-location-activity entry (u_i, l_j, a_k) in rating tensor *T*, HOSVD based technique finds similar entries to u_i, l_j and a_k from reduced-rank matrices and then refers to original tensor *T* to find corresponding rating values. In EFC, we propose to find similarities from 2-dimensional integrated model. To do so, we firstly construct the model from available matrices (details are given in section 4.3.1). Secondly, we apply SVD to extract the similarity matrices and at the end we calculate the value of missing entry using similarity matrices and referring to original rating tensor *T*.

4.3.1 Model Construction

The EFC model is actually produced by integrating partial matrices into a bigger total matrix in which, each part of it corresponds to one of the partial matrices. In the first step, as shown in Figure 5, similarity matrices which are explained in Section 4.2 are inserted into the main diagonal of total matrix. Remember that, the values of similarity matrices are in the range of [0,1] where 0 means absolute dissimilarity and 1 means absolute similarity. Since the values of other matrices are in the range of [0,5] and in order to keep the magnitude of all entries in the same range, we scale all three similarity matrices within [0,5] exploiting min-max normalization method [75]. Using this main model we can construct three different recommendation systems as shown in Figure 15.

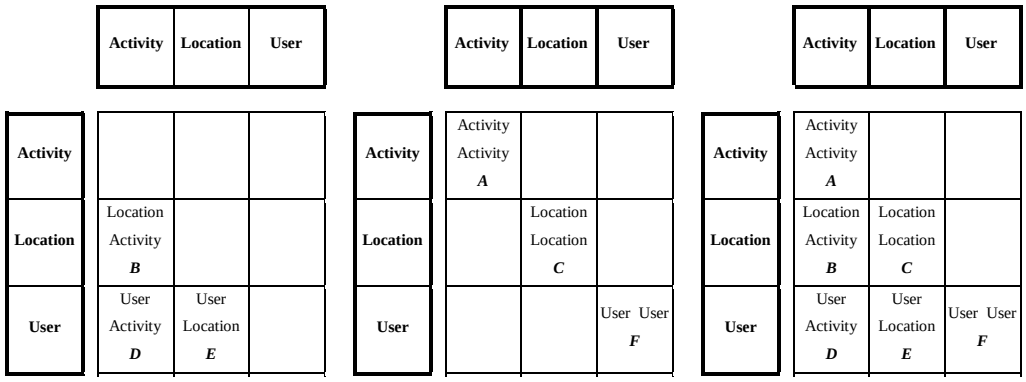

Figure 15. Extended Feature Combination (EFC) models overview (empty sub-matrices contain zero values)

4.3.2 Similarity Calculation

Once we obtain the final integrated matrix, we proceed with applying SVD and finding similarity for a given user, location and activity (u_i, l_j, a_k) in rating tensor *T*. Once SVD is applied to integrated matrix according to Equation 9, matrices U, S and V^T are produced. Based on discussion in Section 3.2, by selecting first r columns of U and first r rows of S and also keeping r greatest singular values from S we may reduce the rank of integrated matrix to r. From now on, we refer to reduced-rank matrices as U_r, S_r and V_r^T. However, the value of parameter r practically affects the total accuracy of model and needs to be determined in advance. Similar to previous method, we can determine the value of r using Equation (3) and maintain 50% of the original data.

As shown in Figure 16, we may think about U_r as row vectors in which columns are representing hidden attributes that reflect the latent relations within data set. Hence, we can trim U_r such that only required information can be selected. If we define $size\_Activity$ to be the number of activities in the data set and select first $size\_Activity$ rows of U_r then, we get a matrix that its rows represent activities and its columns show the latent attributes of data set. Furthermore, we call the matrix U_r $\_Activity$ and use it to calculate similarities between a distinct activity and other activities.

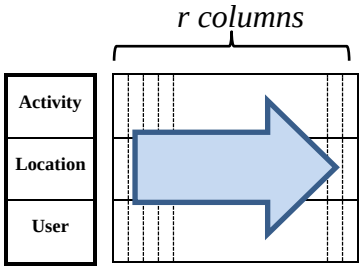

Figure 16. Reduced-rank visualization of *U*

We define $size\_Location$ and $size\_User$ to be the numbers of locations and users in our data set. Similarly, by proper trimming of rows in Figure 16 we get matrices $U_r\_Location$ and U_r_User whose rows represent locations and users correspondingly and columns show the latent attributes of data set. Figure 17 shows the overall schema of trimming process.

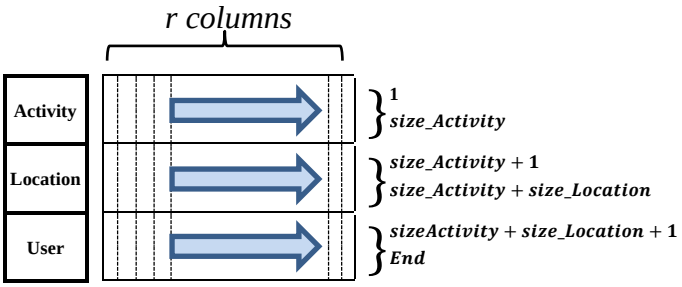

Figure 17. Visualization of matrix trimming

Analogously, we can think about V_r^T as column vectors in which rows are representing latent attributes of data set (Figure 18). By a proper trimming and selecting columns of V_r^T according to size of user, location, and activity we can extract matrices $V_r^T\_Activity$, $V_r^T\_Location$ and $V_r^T\_User$ whose columns are representing activities, locations and users and rows are representing hidden relations of the data set. Trimming process of V_r^T is very similar to Figure 17 with a difference that instead of row-wise selection we trim it column-wise.

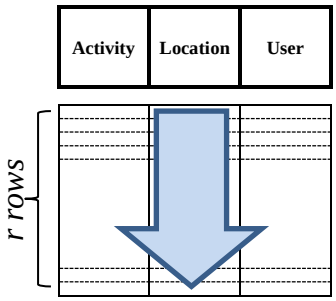

Figure 18. Reduced-rank visualization of V^T

So far, by exploiting U_r and V_r^T and performing trimming on them we have achieved 6 matrices which we can directly use to calculate the required similarity matrices. *Cosine* similarity is a typical and efficient proximity metric which has been reported to outperform other measures in domain of recommendation systems thus, we utilize it at this step as well [77]. Suppose we have chosen $U_r \_ User$ as the matrix that we want to calculate cosine similarity. Column c_i of the similarity matrix shows the similarity between user u_i and all other users in data set. However, we are not interested in all of those users and seeking for the most similar users to user u_i (It means they have the highest values in column c_i of similarity matrix). In order to find similar users conveniently, we sort each column in descending order of similarity magnitude and name it $U \_ User\_sim$ so that, the most similar user to u_i stands at the top of column c_i of matrix $U \_ User\_sim$. We are going to utilize these matrices in the examples below. The trimmed matrices and corresponding sorted similarity matrices together with short comments about each one are listed in Table 4.

Table 4. Similarity matrices explanation

	Trimmed Matrix	Similarity Matrix (Sorted)	Comment
1	$U_r - Activity$	$U - Activity - sim$	Activity-Activity similarity produced from U_r
2	$U_r - Location$	$U - Location - sim$	Location-Location similarity produced from U_r
3	$U_r - User$	$U - User - sim$	User-User similarity produced from U_r
4	$V_r^T - Activity$	$V - Activity - sim$	Activity-Activity similarity produced from V_r^T
5	$V_r^T - Location$	$V - Location - sim$	Location-Location similarity produced from V_r^T
6	$V_r^T - User$	$V - User - sim$	User-User similarity produced from V_r^T

4.3.3 Rating Prediction

In this section we discuss the procedure that predicts the rating value of given entry (u_i, l_j, a_k) in user-location-activity rating tensor *T*. In each step we maintain two indices of

the entry and find similar objects of remaining index using information in Table 4. Remember that, for each of users, locations, and activities we have calculated two similarity matrices which were obtained from U_r and V_r^T respectively.

In the first step, we freeze the index values of location l_j and activity a_k for selected entry (u_i, l_j, a_k). Then, referring to Table 4 we choose similarity matrices at rows 3 and 6 ($U - User - sim$ and $V - User - sim$). Using $U - User - sim$ we select m_1 similar users to user u_i ($\{u_i^m\}$) so that:

$$\forall\, u_i^m \in \{u_i^1, u_i^2, \dots, u_i^{m_1}\}, \qquad T(u_i^m, l_i, a_k) \neq 0 \tag{23}$$

However, instead of traversing the similarity matrix from beginning to end, we only choose a proportion of it as it is discussed in Chapter 5. Once we have found the value of $T(u_i^m, l_i, a_k)$ for all $\{1, \dots, m_1\}$, first prediction is calculated as the weighted average of deviations from the similar neighbor's mean [77]:

$$P_{User}^{U} = \bar{r}_{u_i} + \frac{\sum_{p=1}^{m} \left(T(u_p, l_i, a_k) - \bar{r}_{u_p}\right) * sim(u_i, u_p)}{\sum_{p=1}^{m} |sim(u_i, u_p)|} \tag{24}$$

Similarly, utilizing V_User_sim, we select m_2 similar users to user u_i ($\{u_i^m\}$) so that:

$$\forall\, u_i^m \in \{u_i^1, u_i^2, \dots, u_i^{m_2}\}, \qquad T(u_i^m, l_i, a_k) \neq 0 \tag{25}$$

After finding the values of $T(u_i^m, l_i, a_k)$ for all $\{1, \dots, m_2\}$, another value is calculated as the prediction for given entry (u_i, l_j, a_k) using similarity among users as:

$$P_{User}^{V} = \bar{r}_{u_i} + \frac{\sum_{p=1}^{m} \left(T(u_p, l_i, a_k) - \bar{r}_{u_p}\right) * sim(u_i, u_p)}{\sum_{p=1}^{m} |sim(u_i, u_p)|} \tag{26}$$

In the second step, we freeze the index values of user u_i and activity a_k for selected entry (u_i, l_j, a_k). Then again referring to Table 4 we choose similarity matrices at rows 2 and 5 ($U - Location - sim$ and $V - Location - sim$). The rest of process is similar to previous step except that the value of parameters m_3 and m_4 are different.

Similarly, in step three we freeze the index values of user u_i and location l_j for selected entry (u_i, l_j, a_k) and make predictions using similarity among activities. Table 5 shows the estimated values and parameters that are achieved from mentioned three steps.

Table 5. Estimated values and parameters

	Similarity Matrix	Estimated Value	Size of neighborhood
1	$U - Activity - sim$	$P^{U}_{Activiy}$	m_1
2	$U - Location - sim$	$P^{U}_{Location}$	m_2
3	$U - User - sim$	P^{U}_{User}	m_3
4	$V - Activity - sim$	$P^{V}_{Activity}$	m_4
5	$V - Location - sim$	$P^{V}_{Location}$	m_5
6	$V - User - sim$	P^{V}_{User}	m_6

The final step combines 6 predictions that are calculated from previous steps. We may have a weighted combination method which assigns separate weights for each individual prediction. However, for simplicity we assign same weights for all predictions. Though, the final prediction for a given entry (u_i, l_j, a_k) in tensor T can be calculated as following equation:

$$\hat{T}(u_i, l_j, a_k) = \frac{\boldsymbol{W}.\boldsymbol{Prediction}}{\sum_{i=1}^{6} w_i} \tag{27}$$

In the Equation (27), $\boldsymbol{W}$ is weight vector and $\boldsymbol{Prediction}$ is vector of predictions that are shown in second column of Table 5. The operation in enumerator is matrix dot product.

In the rest of this section, we present three different recommendation models as the variations of EFC.

4.3.4 Feature Combination Using Collaborative filtering

In this model, we aim to reconstruct a model that utilizes only the data which is extracted from rating tensor T. In other words, the condition is to integrate those matrices from Figure 15.a that reflects the property of users' rating and not any contextual information about users, locations, or activities. Main idea for this reconstruction is to include the impact of only rating on prediction, without using the information coming from resources other than users' feedback.

As discussed in the Section 4.1, we have only three matrices Location-Activity (*B*), User-Activity (*D*), and User-Location (*E*) in which satisfy the condition above. To accomplish the reconstruction, matrices with all zero values also have to be inserted into the model. Integration of those matrices is depicted in Figure 15.a.

We construct a sample model using the rating matrix of Figure 10 such that we only insert matrices which are achieved from rating values into the final model. Once the SVD is applied to the model and rank of it is reduced to 2, matrices in lines 1, 2, 3, and 5 of Table 4

and their corresponding similarities in Table 5 are calculated. Figure 19 illustrates the integrated matrix and Figure 20 presents the matrices obtained from applying SVD and rank reduction. Remember that we sort the similarity matrices in descending order as are depicted in Figure 21 and Figure 22 as forms of indices and values.

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
3.4	3.6	4	1.75	0	0	0	0	0	0	0	0	0	0
2.4	2.4	1.5	1.5	0	0	0	0	0	0	0	0	0	0
2.67	2	2.6	2.67	0	0	0	0	0	0	0	0	0	0
1.5	0	0	3	0	0	0	0	0	0	0	0	0	0
2	1	3.5	2	0	0	0	0	0	0	0	0	0	0
2.5	2	3	2.5	2.75	2.5	1	2.5	3.33	0	0	0	0	0
2.75	2	3	2.67	3	2.67	1.5	3	2.67	0	0	0	0	0
3	1.67	2.33	2	3.25	1.25	2.25	0	3	0	0	0	0	0
2.5	3.5	3.337	1.5	4	2.5	2.75	1	1.75	0	0	0	0	0
1.75	3.33	3.67	2	3	1.67	3.5	0	1.67	0	0	0	0	0

Figure 19. Collaborative filtering based combination model

-0.406	0.2522			-0.4078	-0.4195			-0.2643	-0.6713			-0.3718	0.4363
-0.423	0.263			-0.3931	-0.3892			-0.1735	-0.3765			-0.2482	0.316
-0.368	0.1665			-0.47	-0.3219			-0.1058	-0.3095			-0.2553	0.2509
-0.4443	0.127			-0.2974	0.0773			-0.0849	-0.0648			-0.1541	0.2522
-0.4087	0.0479							-0.1947	-0.3589			-0.2848	0.3822

a) U-User **b) V-Activity** **c) U-Location** **d) V-Location**

Figure 20. Reduced rank matrices

2	1	1	3	4
3	3	2	5	3
4	4	4	1	1
5	5	5	2	2
1	2	3	4	5

a) Sorted index of U_User_sim

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

***b*) Sorted index of $V\_Activity\_sim$**

3	1	1	5	2
2	5	2	2	1
5	3	5	1	3
4	4	4	3	4
1	2	3	4	5

c) Sorted index of $U\_Location\_sim$

2	5	1	5	2
5	1	2	2	1
3	4	5	1	4
4	3	4	3	3
1	2	3	4	5

d) Sorted index of $V\_Location\_sim$

Figure 21. Similarity indices of CF model

0.999	0.999	0.9914	0.9893	0.987
0.9914	0.9914	0.9914	0.987	0.9529
0.9618	0.9616	0.9893	0.9618	0.9051
0.9051	0.9049	0.9529	0.9616	0.9049
1	1	1	1	1

a) Sorted value of U_User_sim

0.9998	0.9998	0.9839	0.6564
0.9802	0.9839	0.9802	0.5108
0.4942	0.5108	0.6564	0.4942
1	1	1	1

***b*) Sorted value of $V\_Activity\_sim$**

0.9938	0.9969	0.9985	0.9985	0.9938
0.9814	0.9912	0.9969	0.9912	0.9538
0.9633	0.9814	0.9633	0.9472	0.9273
0.9472	0.9538	0.9273	0.9055	0.9055
1	1	1	1	1

c) Sorted value of $U\_Location\_sim$

0.9992	0.9997	0.9961	0.9958	0.9997
0.9979	0.9992	0.9918	0.9931	0.9979
0.9961	0.9931	0.9882	0.9876	0.9958
0.9876	0.9918	0.9699	0.9699	0.9882
1	1	1	1	1

d) Sorted value of $V\_Location\_sim$

Figure 22. Similarity values of CF model

In order to illustrate the model to predict the value of first randomly selected entry $a_1 = (2,1,2)$ with original value of 3, we freeze the index of $location = 1$ and $activity = 2$ then find the most similar user (the size of neighborhood in this case is 1) to $user = 2$ which according to column 1 of Figure 21.a is $user = 1$.

$$P_{User}^{U} = 2.5833 + \frac{(3 - 2.5333) * 0.9999}{|0.9999|} = 3.05$$

In the second prediction we freeze the index of $user = 2$ and $activity = 2$ then find the most similar location to $location = 1$. However, we can find the similarity from two matrices in Figure 21.c and Figure 21.d which is $location = 3$ and $location = 2$. Hence, we obtain the predictions as:

$$P^{U}_{Location} = 3.2222 + \frac{(1 - 2.4375) * 0.9938}{|0.9938|} = 1.76$$

$$P^{V}_{Location} = 2.5625 + \frac{(2 - 2.0625) * 0.9992}{|0.9992|} = 3.16$$

Finally, we keep the index of $user = 2$ and $location = 1$ and find the most similar activity to $activity = 2$ utilizing second column of Figure 21.b which is activity 1 and $T(2,1,1) = 3$.

$$P^{V}_{Activity} = 2.5625 + \frac{(3 - 2.5) * 0.9998}{|0.9998|} = 3.06$$

The final prediction for entry $a_1 = (1,2,2)$ is calculated from partial predictions that are produced from previous steps as formulated in Equation (27).

$$\hat{T}(2,1,2) = \frac{[1\ 1\ 1\ 1].[3.05\ 1.76\ 3.16\ 3.06]^T}{4} = 2.76$$

4.3.5 Feature Combination Using Contextual Information

In the second model of EFC, we are interested in considering the effect of contextual information on accuracy of predicted values as well. In this case contextual information is such information that is not achieved from rating matrix. In contrast, it is related to contents of users, locations, and activities which reflect their intrinsic characteristic. That information is comprised of *User-User*, *Location-Location*, and *Activity-Activity* similarity matrices which were discussed in detail before.

As the model reconstruction in Figure 15.b presents, contextual information is inserted as the main diagonal and similar to other models zero is inserted to the other components of the model in order to construct the big integrated matrix. The procedure of finding similarity matrices is very similar to what explained in Section 4.3.2.

As discussed in contextual model definition, in the following example we merely insert similarity matrices into the combination model. Similar to the previous method, in each step index values of 2 dimensions are frozen and similar entries are found using the 3rd dimension of randomly selected entry from tensor T. Note that, regarding to Table 5 and the construction of model for contextual information, we only have 3 similarity matrices that are available in rows 1, 2, and 3 of Table 5. Calculation is analogous to that explained in CF model hence we only present value of each prediction and final estimation of given random values.

$$\hat{T}(2,1,2) = \frac{[1\ 1\ 1].[2.5833\ 3.2222\ 2.5625]^T}{3} = 2.79$$

4.3.6 Hybrid Feature Combination

In previous models, each of integrated data has specific properties which both are informative and reveal us different aspect of data set. Proposed hybrid model corresponds to combining both of the models explained above. In this case, we combine contextual information with the data extracted from rating values.

In this model we insert Location-Activity (*B*), User-Activity (*D*), and User-Location (*E*) matrices into the output from previous step. As Figure 15.c illustrates, we still have 3 missing sub matrices. One solution to complete the missing parts is to use transpose of Location-Activity (B^T), User-Activity (D^T), and User-Location (E^T) from model 1 in Figure 15.a. However, if we want to treat fairly, we should not utilize them twice even with different format. Hence, we insert zeros into missing parts of model which completes the matrix and makes it lower triangular.

We demonstrate the hybrid feature combination model with an example. The typical data set which is introduced in Figure 10 is used in this example. It is a real sub tensor of the original data and the similarity matrices are also available as shown in Figure 23.

1	0.9	0.5	0.7	0.55
0.9	1	0.7	0.4	0.3
0.5	0.7	1	0.3	0.1
0.7	0.4	0.3	1	0.8
0.55	0.3	0.1	0.8	1

a) User-User similarity

1	0.6	0.5	0.1	0.3
0.6	1	0.7	0.75	0.65
0.5	0.7	1	0.3	0.1
0.1	0.75	0.3	1	0
0.3	0.65	0.1	0	1

b) Location-Location similarity

1	0.6	0.9	0.7
0.6	1	0.2	0
0.9	0.2	1	0.1
0.7	0	0.1	1

c) Activity-Activity similarity

Figure 23. Similarity matrices

Additionally, the model requires three more partial matrices to complete the final integrated matrix as displayed in Figure 15.c which can be easily calculated from tensor T as explained in Section 4.1. In order to illustrate execution of the model, prior to calculating them, we replace values of randomly chosen nonzero entries $a_1 = (5,1,2)$, $a_2 = (3,3,3)$ and $a_3 = (1,1,2)$ in T with zero. Calculated matrices are shown in Figure 24.

3.4	3.6	4	1.75
2.4	2.4	1.5	1.5
2.67	2	2.6	2.67
1.5	0	0	3
2	1	3.5	2

a) Location-Activity

2.5	2	3	2.5
2.75	2	3	2.67
3	1.67	2.33	2
2.5	3.5	3.337	1.5
1.75	3.33	3.67	2

b) User-Activity

2.75	2.5	1	2.5	3.33
3	2.67	1.5	3	2.67
3.25	1.25	2.25	0	3
4	2.5	2.75	1	1.75
3	1.67	3.5	0	1.67

c) User-Location

Figure 24. Additional extracted matrices

Once we get all partial matrices, we proceed by constructing the model according to Figure 15.c. As explained in model construction, similarity matrices are located in main diagonal and other matrices are inserted into the lower triangle of model. Applying SVD to the model and reducing its rank to 3 (to keep 50% of original data) from one side and trimming it together with calculation of Cosine similarity from other side, results in getting matrices that are shown in second column of Table 4.

In order to make prediction for the value of $a_1 = (2,1,2)$, we freeze the index of $location = 1$ and $activity = 2$ and find similar users referring to Figure 25. As shown in column 2 of part a, the most similar user to $user = 2$ is user number 3. Additionally, using part d the most similar user to $user = 2$ is also user 1. Note that values of parameters m_1 to m_6 are set to 1. Predictions from users are calculated as:

$$P^{U}_{User} = 2.5833 + \frac{(3 - 2.3077) * 0.9998}{|0.9998|} = 3.28$$

$$P^{V}_{User} = 2.5833 + \frac{(3 - 2.5333) * 0.9995}{|0.9995|} = 3.05$$

2	3	2	5	4
3	5	5	3	3
5	4	4	2	2
4	1	1	1	1
1	2	3	4	5

a) Sorted index of U_User_sim

5	3	4	3	1
2	4	2	2	2
3	1	1	1	3
4	5	5	5	4
1	2	3	4	5

***b*) Sorted index of $U\_Location\_sim$**

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

c) Sorted index of $U\_Activity\_sim$

3	1	1	5	4
2	3	2	3	3
4	4	4	1	1
5	5	5	2	2
1	2	3	4	5

d) Sorted index of V_User_sim

5	4	2	2	1
3	3	4	3	3
2	1	1	1	2
4	5	5	5	4
1	2	3	4	5

e) Sorted index of $V\_Location\_sim$

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

f) Sorted index of $V\_Activity\_sim$

Figure 25. Indices of similarities

In the second step, we freeze the index of $user = 2$ and $activity = 2$ and find similar locations to $location = 1$. According to part b and e of Figure 25, the most similar location to $location = 1$ is 5 in both parts. Since $T(2,5,2) = 0$ second similar are chosen which are 2, 3 respectively. The calculations are given by:

$$P^{U}_{Location} = 3.2222 + \frac{(2 - 2.0625) * 0.8444}{|0.8444|} = 3.16$$

$$P^{V}_{Location} = 3.2222 + \frac{(1 - 2.4375) * 0.7533}{|0.7533|} = 1.79$$

Similarly, by freezing the index value of $user = 2$ and $location = 1$ we find the most similar activity to $activity = 2$ which in parts c is 1and in part f is 4. Since $T(2,1,4) = 0$ second similar in part f is chosen to be 1 hence, predictions are as follows:

$$P^{U}_{Activiy} = 2.5625 + \frac{(3 - 2.5) * 0.9444}{|0.8444|} = 3.06$$

$$P^{V}_{Activiy} = 2.5625 + \frac{(3 - 2.5) * 0.8144}{|0.8144|} = 3.06$$

The final prediction for $a_1 = (2,1,2)$ as discussed before is calculated from the mean of all predictions as follows:

$$\hat{T}(2,1,2) = \frac{[1\ 1\ 1\ 1\ 1\ 1].[3.28\ 3.05\ 3.16\ 1.76\ 1.79\ 3.06\ 3.06]^{T}}{6} = 2.97$$

CHAPTER 5

EXPERIMENTAL RESULTS AND EVALUATION

This chapter presents the experiments that were carried out in order to evaluate the performance of the system. First, the datasets that are used for evaluating the system are described. Then, the used metrics are specified. Next, the evaluation of the hybrid system is performed including comparisons with two different methods and data sets. Finally, results are presented with graphical charts.

All the experiments are performed on a 2.53 GHz PC machine with 4 GB of main memory. Programs are written in MATLAB R2009a [72] and tensor dimensionality reduction part is implemented using Sandia tensor toolbox [66][67].

5.1 Data sets

In this thesis two different datasets are used in order to evaluate the proposed system. These are the CLAR data sat that is introduced in [45] and the one which acquired from an online project [55] which is administered by Symeonidis [56].

5.1.1 CLAR Data set

The first data set that is used in this thesis is gathered by Microsoft Research Asia. Pre-processed data set is available online and can be downloaded from Microsoft Research Asia website [54]. The data set is collected from a web-based application over 2.5 years and contains over 12,000 GPS trajectories. In this system, each user is equipped with a GPS installed tool (GPS Navigator, smart phone) during visiting Beijing city in China. For each location that is visited, users may insert comments about that place and available five activities that are done in that location. These five activities are food and drink, shopping, movie and shows, sports and exercise, tourism and amusement. Thus, these comments are processed and together with location information constitute Location-Activity matrix, whose rows are locations and columns are activities each entry shows the frequency of doing an activity. The raw data is organized in three tables which are explained as follows:

- **Location-Activity:** Contains ratings of all users for different activities. Values are non-negative integers and zero value in an entry denotes that there is no rating

available in a location for doing activity in corresponding row and column. There are 167 locations and 5 activities.

- **Location-Feature:** Contains frequency of several POIs (Point of Interest) for any location which is extracted from city's yellow page. Entries are non-negative integers and there are 13 features available.
- **Activity-Activity:** It is a similarity matrix whose entries are real values in interval [1,-1] and shows correlation within 5 activities.

The other data set is in a row format and the table extraction methods are discussed in Chapter 3. Quantity of User, Location, and Activity are 149, 438, and 112 correspondingly.

- Location-Activity
- Location-User
- Activity-User
- User-User
- Activity-Activity
- Location-Location

The last three data tables in the previous list show the similarity within user, activity, and location entities.

5.1.2 Geosocial2 Data set

The experimental evaluation of the EFC approach is conducted on Geosocial2 data set which is available online in [55]. Since the data set is used for the first time in literature, we present more details on the data set. Geosocial2 is gathered form ratings that are given by 149 users to various 112 social activities. Ratings are submitted to system for performing the activities in 438 locations which in turn are located in some cities of Greece.

It consists of "Activities", "Places" (coordinate and name of them), "Users" (user profiles), Paths (users' friendship network) matrices and finally the most important one, "Check-ins" table which shows the relation between a user, activity, location, rating and the time stamp that is assigned to occurrence of it. We organize "Check-ins" matrix in a 3-order tensor T so that, each entry $T(x,y,z)$ shows the rating that user z assigns to activity y in location x. Since similarity matrices are not directly available, we have to extract them from data set.

5.2 Prediction Evaluation Metrics

There are different approaches to evaluate the performance of an information retrieval system. Since In this thesis our task is mainly value prediction, to compare the results numerically, we have used *Root Mean Squared Error (RMSE)* and *Mean Absolute Error (MAE)* which both measure difference between observed values (original values) and estimated values.

$$MAE(O,E) = \frac{1}{n}\sum_{i=1}^{n}|o_i - e_i| \tag{28}$$

$$RMSE(O,E) = \sqrt{\frac{\sum_{i=1}^{n}(o_i - e_i)^2}{n}} \tag{29}$$

In both Equations (28) and (29), *O* denotes the original value and *E* is the value which is calculated by any prediction technique.

As far as the rating values are distributed in a range, we also can use *log* function to squeeze the interval to observe differences between original values and estimated values. As well as *MAE* and *RMSE*, we apply different bases from log_2 to log_5 to better compare the results.

In addition, we have proposed an evaluation method which we call it *Abstraction* method. In this technique, we partition the nonzero values of Location-Activity matrix to *n* clusters using *k-means* clustering algorithm. In this algorithm, *n* random points are selected as initial mean values randomly within the ratings values then, each point (rating value in this case) is assigned to a cluster according to shortest value of *Euclidean* distance from each mean values. In this step for each cluster, mean value is calculated and assigning points to each cluster is performed iteratively, until no point assigned to a new cluster.

In this abstraction method, instead of working with rating values we have examined the cluster that each value belongs to. Note that, the clusters are ranked in a single dimension. Instead of computing error between original value and predicted value, we find the distance between the clusters that original value belongs to and the cluster that predicted value falls into it. Since both data sets' rating intervals are a standard 1 to 5 rating systems, we determine 5 for value of *n* clusters.

In order to clarify this procedure, we demonstrate the calculation for the example of previous Section. The nonzero values of matrix *X* in Equation (8) are clustered to 3 clusters using k-means and corresponding clusters are:

$$C1 = \{1,1,2,3,4,17,17\} \quad C2 = \{53,53\} \quad C3 = \{76,82\} \tag{30}$$

According to Equation (30), validation data $x_{11}=1$, $x_{22}=17$ and $x_{33}=53$ belongs to *C1*, *C1* and *C2* and regarding to Table 1 the predicted values for them are *19.5*, *18.5* and *43.16* which indicates that new clusters for predicted values are *C1*, *C1* and *C2*. As an example, let's assume that for a given validation data x_{ij} the original cluster is *C1*, our predicted value falls into cluster *C3* and the predicted value in work [8] falls into *C2* thus, the MAE error can be calculated consequently as $|1-3| = 2$ for proposed method and $|1-2| = 1$ for CLAR.

5.3 Recommendation Evaluation Metrics

In a typical recommendation system items are recommended to users. Since in our model we have both activities and locations, our system can be used to recommend activities at certain locations, or locations for certain activities. We simply predict the rating value of user-location-activity triple, if it is unknown, and then generate either activity or location recommendation if that rating value is greater than some threshold.

To assess the quality of the recommendations generated by the methods introduced in this paper, we employed precision and recall which are standard and widely used metrics in information retrieval. The way we measure them for each method is as follows:

- First, we randomly pick 10% of the non-zero entries from rating tensor and set their values to be zero.
- Then, we use one of our methods and predict their values.
- For each predicted value we determine the followings:

I. *True positives (TP)*: if both the original value and its predicted value are greater than the threshold then that means we have made correct recommendation.
II. *False positives (FP)*: if the original value is less than the threshold, but the predicted value is greater than it, then that means we have made an incorrect recommendation.
III. *True negatives (TN)*: if both the original value and the predicated value are less than the threshold, then that means we did not make recommendation and it was correct.
IV. *False negatives (FN)*: if the original value was greater than the threshold, but the predicted value was less than it, then we have missed the recommendation that we should have done.

Using these values, precision and recall are calculated as combination of TP, FP, and FN as given in Equations (31) and (32).

$$Precision = \frac{TP}{TP + FP} \tag{31}$$

$$Recall = \frac{TP}{TP + FN} \tag{32}$$

5.4 Evaluation Method

In order to evaluate our results, we should separate test data and train data in both data sets. As a result, well-known *k*-fold cross-validation method is used to partition the whole data set into a training data set and a validation data set. To be able to apply this validation technique we have assumed that our data set is accurate. Then, we have set $(1/k)^{th}$ of nonzero entries of the Location-Activity matrix to zero. Afterwards, we have applied our approach on the data set to predict the values of those entries. The *k* results from the folds then can be averaged (or otherwise combined) to produce a single estimation.

Evaluation of results in this thesis is actually divided to two phases. In first phase, we compare our results with a state-of-the-art method in Geo-Activity recommendation named CLAR, which is proposed by Zheng in [45]. In order to calculate prediction accuracy, we use MAE and RMSE and the proposed abstraction method.

In second phase, we will reduce a Tensor-based recommendation system [61] to 2-D matrices mode and exploit our model to make prediction in matrix mode instead of tensor mode. In addition to metrics mentioned in phase 1, we will apply the discussed *Log* method as well.

5.5 Experiments for IFC

As we discussed in Section 4.3, in order to prepare training and validation data, we have used *k*-fold cross-validation. As it is used usually, we put $k = 10$ that is, in each fold of execution, we select *10%* of nonzero entries in location- activity matrix randomly and put them as validation set, remaining part is the training set and it is used to construct the prediction model. As discussed in Section 3, by using additional data, we implement the proposed method to predict the rating value of all entries in location-activity matrix. Since for validation data we have both observed value and predicted value, we can calculate *RMSE* and *MAE* for this fold. Therefore, in each loop we acquire a value for *RMSE* and *MAE*. At the end of last loop we sum up results for all folds and calculate the mean value of corresponding error terms. Obviously, all steps of 10-fold cross-validation were also applied to CLAR. Each column in Figure 26 shows the mean value of *MAE* for 10 folds. In order to have a comprehensive view of results we run the application for 10 times and put the mean value of *MAE* in a new column.

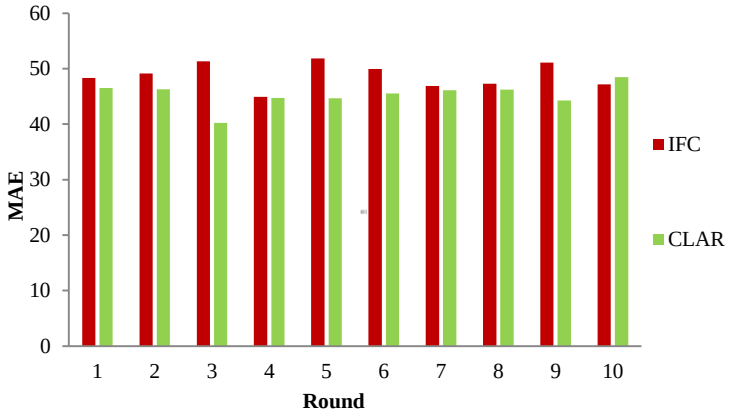

Figure 26. MAE values for proposed work vs. CLAR

Figure 27 shows the same comparison of RMSE for IFC and CLAR. Like MAE, we run the application for 10 times that each column shows mean value of RMSE for 10 folds.

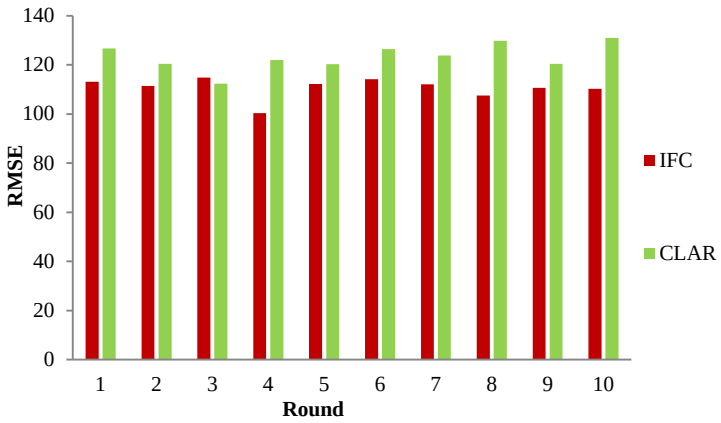

Figure 27. RMSE values for proposed work vs. CLAR

Figure 28 shows the results of MAE when we apply the abstraction technique. Similar to the previous method each column shows mean value of MAE for 10 folds. We run application for 10 times to show a comprehensive view of results.

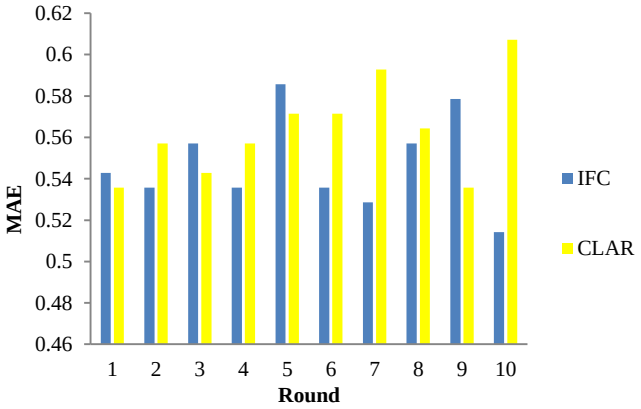

Figure 28. MAE values for proposed work vs. CLAR with applying abstraction

Finally, Figure 29 shows the mean value of RMSE for 10 folds in one column. For the same reason the application is run for 10 times and each column shows mean value of RMSE in each time of execution.

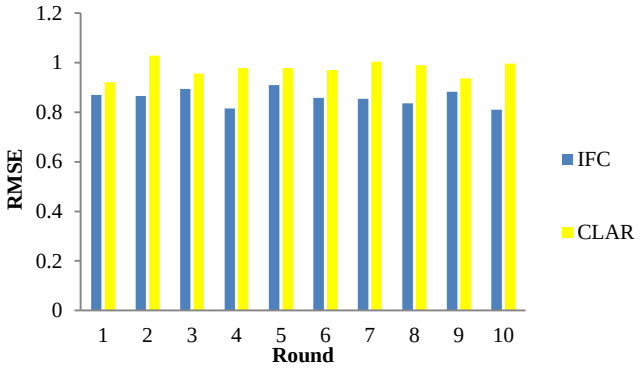

Figure 29. RMSE values for proposed work vs. CLAR with applying abstraction

5.5.1 Parameter Tuning

Since each data set has its own characteristics, it is not possible to have one set of fixed parameters that works for all. In our work, we have parameterized possible number of top similar locations and top similar activities.

Due to the fact that value of parameters *m* and *n* which denote the most similar values of rows and columns affect the errors, we have probed different values of these parameters to find the optimal values that reduce the *RMSE*. For each of *m* and *n* we choose values from 1 to 5 thus, we have obtained 25 combinations of them that can be organized in 5 diagrams. We have also implemented it for *RMSE* without abstraction and with abstraction as discussed in the Section 5.2.

As illustrated in Figure 30, values of *m=1* and *n=3* when results are not abstracted lead to minimum *RMSE*.

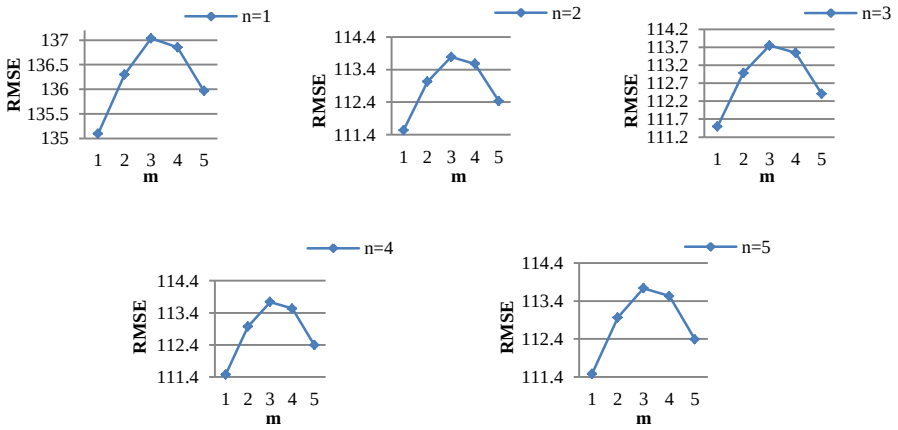

Figure 30. *RMSE* vs. parameter *m* without abstraction

Similarly, Figure 31 shows that m=4 and n=4 are the best values when the abstraction is applied, that gives the minimum RMSE.

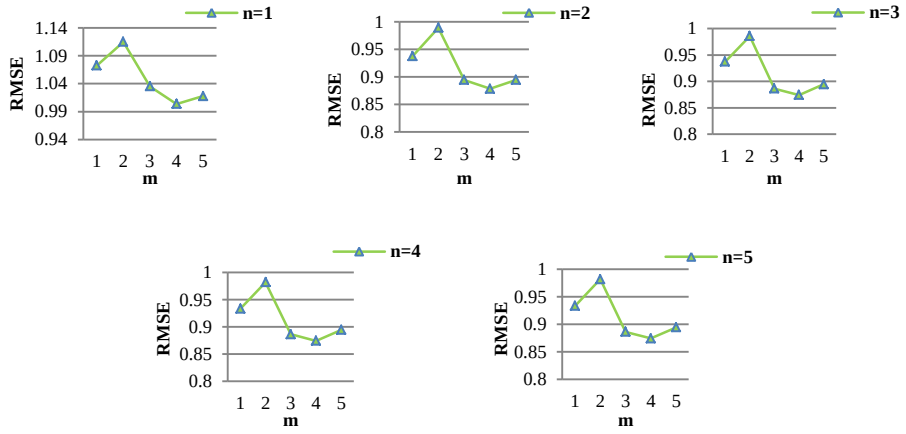

Figure 31. *RMSE* vs. parameter *n* with abstraction

In this data set for a fixed *n*, RMSE increases up to *m=3*, and then it starts to decrease. However, for even larger values of *m* it does not reach to the level for *m=1*. Therefore, *m=1* seems like the most appropriate choice. Although for different values of *n* slightly different results are obtained, except for 1 they are quite close to each other. Moreover, among them *n=3* gives the smallest *RMSE* values.

On the other hand, when the abstraction is used, for a fixed n, RMSE shows rather fluctuated behavior for increasing *m*. In all the cases we have tested, *m=4* gives the smallest *RMSE* value, and similar to the previous case except for 1, for all other *n* values results are very close to each other, and *n=4* is the smallest among them.

Considering the size of Location-Activity matrix (167 by 5), searching for similar entries more than 5 would not be feasible. It is not possible to interpret why these values produce the best *RMSE* values, but, it is quite clear that it is directly dependent to the data set (the matrix).

5.6 Experiments for EFC

In this phase, we evaluate the efficiency of merging model which was introduced in Chapter 4. As explained before, our aim is to reduce a Tensor-based prediction to a matrix-based prediction. To achieve this goal, we introduced an extension to prior feature combination which, inject several matrices into a single model and utilize it to predict missing values.

Same as previous phase, we use *k*-fold cross validation technique to create a training data and test data. We determine value of *k* to be 10. In each fold we substitute corresponding values of test data with zero value. As discussed in Chapter 4, when we insert zero to one entry of tensor, two entries are set to zero from our model.

5.6.1 Selecting Neighborhood

Selecting proper neighborhood set is tricky task which potentially influences the accuracy of prediction. One solution is to choose *top-N* neighborhood in which neighbors with the highest similarity to active entry (user, location, or activity) is selected to predict the target rating [78]. However, determining the magnitude of *N* is crucial since if it is too low, it cannot be a precise prediction and if it is too high, the noise will affect the result negatively.

An alternative solution proposed in [13] which chooses neighbors with a threshold value of similarity. For instance one strategy is to add such entries to neighborhood set that the value of similarity (correlation) is greater than 0.7 to active entry. Another simple method which mostly is utilized in small data sets is to set a threshold to the percent of similarities that are going to determine neighborhood. In this case for example we can seek top 10% of sorted similarity matrix to construct neighborhood set.

When there is no neighbor for the defined boundaries, zero value is set as the prediction value. Figure 32 depicts the influence of neighborhood size on MAE for all methods. In HOSVD, for a small size of neighborhood MAE is high but as the size is increased, error is decreased and reaches to a minimum value. Collaborative filtering shows an interesting result since it begins with high MAE for small size of neighborhood an soon after it reaches to 5, MAE shows a steady trend to the size of neighborhood.

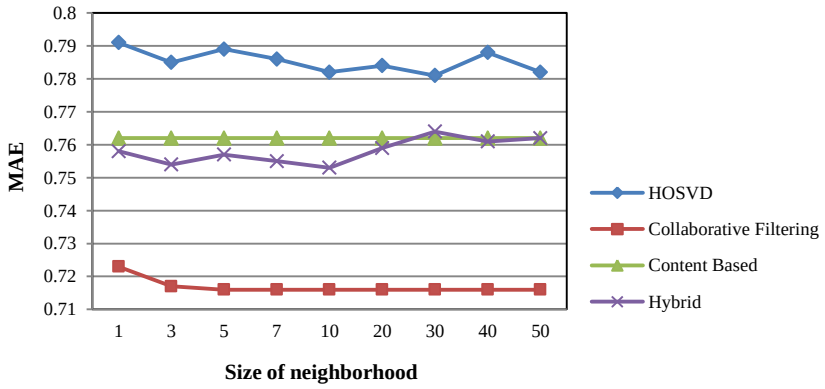

Figure 32. Neighborhood size vs. MAE

In the context based model, in which we use the similarity matrices to construct the model, MAE tends to be steady by changing the size of neighborhood. Finally, Hybrid model shows a slightly fluctuations by increasing the size of neighborhood and it reaches to the lowest value of MAE in size=10 and after that point it keeps an ascending move. It can be the influence of extra noise in calculation as the size of neighborhood is increased.

5.6.2 Weight of Combination

In the prediction process, once the neighbors are determined, we should somehow combine the neighbors' ratings to produce the target prediction. A trivial method uses an average neighbors' ratings so that, rating of user a for item b ($P_{a,b}$) is calculated using ratings given by n neighbors of a for item b ($P_{j,b}$) as shown in Equation (33).

$$P_{a,b} = \frac{\sum_{j=1}^{n} P_{j,b}}{n} \tag{33}$$

However, this is non-personalized method which does not consider the correlation among users, location, or activity. In order to utilize the correlation among users, when it is available, in a better way, use of weighted average is proposed in [37], which takes similarity into consideration as given in Equation (34). In this formula, $w_{a,j}$ is the similarity among user a and its neighbor j.

$$P_{a,b} = \frac{\sum_{j=1}^{n} P_{j,b}.w_{a,j}}{\sum_{j=1}^{n} |w_{a,j}|} \tag{34}$$

Nevertheless, there might be subjective factors in real world problems. There are some users that always give high scores whatever they rate for. In contrast, some other users tend to give very low scores. Therefore, the deviation of ratings from rating mean of each entry should be considered as an effective factor to prediction step. In [77], deviation from average rating of all entries (either user or activity or location) is calculated and incorporated in the prediction as given in Equation (35) in which, $\bar{r}_a$ and $\bar{r}_j$ are average rating of user a and its neighbor j respectively.

$$P_{a,b} = \bar{r}_a - \frac{\sum_{j=1}^{n} (P_{j,b} - \bar{r}_j).w_{a,j}}{\sum_{j=1}^{n} |w_{a,j}|} \tag{35}$$

We have made experiments with all three equations discussed above. We employ Equation (35) to combine the results obtained from each neighborhood, since it produced the best accuracy result.

5.6.3 Other Parameters

The rank of SVD or HOSVD defines the ratio to maintain the percent of original data as shown in Equation (3). Hence, by changing the percent of the data we control the rank of dimension reduction in each model. As illustrated in Figure 33 we analyze the effect of the percent to MAE. HOSVD model shows the optimum result at *10%* of original data and the performance drops as the percent is increased. Together with increasing the percent of original data to maintain, actually some noise is included with it and cause the increasing trend of MAE.

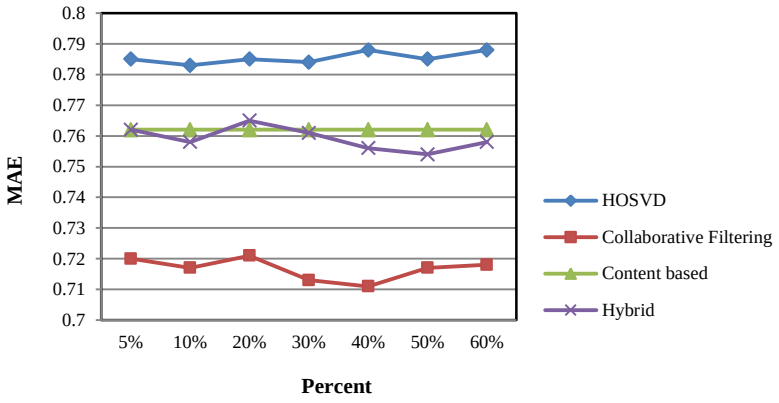

Figure 33. Percent of the data maintained in dimension reduction vs. MAE

Collaborative filtering and hybrid models show slightly similar affect to the magnitude of rank. Both start from an initial MAE for lower ranks and after some fluctuation reach to a minimum value of MAE and take an increasing trend as the rank gets larger. Similar to the same trend in size of neighborhood given in Figure 32, context based model shows a steady trend and changing the rank of reduced-dimension data has no effect on this model.

As described in Section4.3.6, in order to construct EFC model we utilize the $activity - activity$ similarity matrix. Moreover, as also discussed about before, there are different distance metrics that are used to calculate distance between words within WordNet [74]. Utilizing various metrics yields different results for similarity matrix which in turn influences the efficiency of final predicted results. As presented in Figure 34, among several distance metrics we experimented with to find similarity among activities, "*Wu And Palmer"* has better results in proposed EFC.

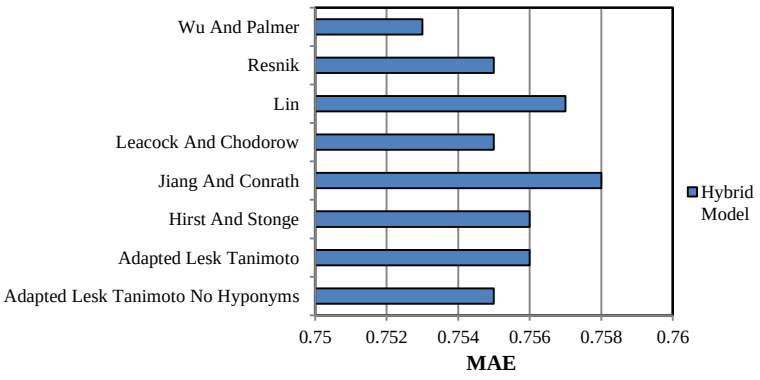

Figure 34. Distance metrics of WordNet vs. MAE in Hybrid model

5.6.4 Prediction results

Table 6 presents the comparison between different models that we discussed for MAE, RMSE, and execution time. Running time can also be a quantitative factor to assess the response time of each technique. However, is divided to two parts in which, first one is the time of preparation for the task of prediction and the second one is time of actual prediction and top-N recommendation. Note that, in each fold of execution we randomly select 10% of nonzero values of rating tensor (which in this work is 82) and replaced them with zero. That means it is the time required for predicting not only one entry but all those entries selected randomly from data set.

Table 6. Comparison between models according to MAE, RMSE, and time of execution

	MAE	RMSE	Preparation Time (sec)	Prediction Time (sec)
Tensor Based	**0.767**	**0.958**	**0.83**	**1.12**
Collaborative Filtering	**0.72**	**0.927**	**0.43**	**0.84**
Contextual Information	**0.764**	**0.945**	**0.37**	**0.78**
EFC	**0.755**	**0.955**	**1.12**	**1.78**

5.6.5 Recommendation results

As explained above, recommendation in top-*N* neighbor system is determined using a threshold. In this work, we have examined the values of 2, 3, and 4 for threshold. The impact of each value on precision and recall are shown in Figure 35 to Figure 37.

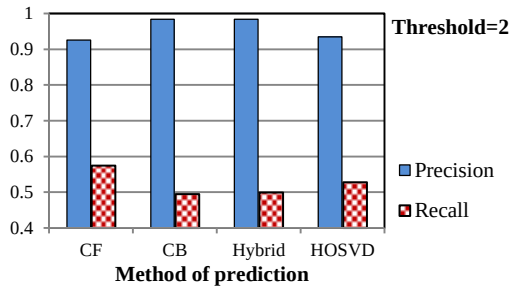

Figure 35. Impact of threshold on precision and recall (threshold=2)

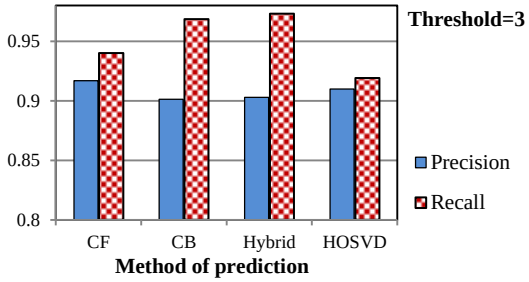

Figure 36. Impact of threshold on precision and recall (threshold=3)

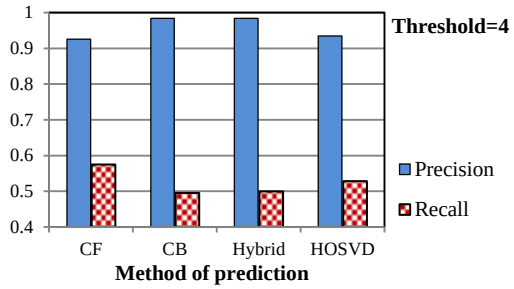

Figure 37. Impact of threshold on precision and recall (threshold=4)

CHAPTER 6

DISCUSSION AND CONCLUSION

In this thesis, a new approach has been proposed for combining additional information into main sparse data set for making recommendation. This idea has been applied to geo-spatial data set for activity-location recommendation system. Activity correlation data and location feature data has also been added into the system in order to improve the accuracy for predicting the missing values of the sparse location-activity data set. The same problem has already been investigated in [46]. Unlike that work, which aimed to complete the whole matrix, we have used low-rank approximation approach of SVD in order to reply recommendation requests when they arrive. Since the original matrix has been reduced to smaller matrices, its memory requirement and construction times are much less than the method of [46].

Furthermore, we use a different method for prediction, which aims to make prediction only cell-wise. This leads to further time efficiency. Indeed, both approaches have their own pros and cons. In large data sets with low recommendation requests our method is more applicable. Moreover, through some experiments, we have also shown that the accuracy values of our approach are better than the one obtained in [46].

Our main idea can be summarized as combining several matrices into single big matrix, and the applying SVD for dimensionality reduction, and finally looking for similar entries for an entry whose rating value is being predicted. Since an entry has three dimensions in 3-D data structure, this similarity search operation has been performed on different domain corresponding to the different parts of the low-rank reduced matrices obtained after SVD, which is the main reason for constructing the results efficiently. Moreover, since the information loss is very little due to 3-D to 2-D reduction process, the accuracy of the prediction is also very high.

HOSVD based approach has already been effectively used for 3-D rating data. However, our experiments show that pure collaborative filtering on the 2-D reduced model is even more effective and efficient than HOSVD based solution. Moreover, the model is also suitable for content based recommendation systems if only similarity matrices are utilized. Although it is not better than collaborative filtering method, the results for content-based method is surprisingly very good as well. Finally, our method allows combining 3-D data together with additional feature matrices very easily. Although we had anticipated even

better results by combing these additional features, at least on our data set the accuracy values were obtained between pure collaborative filtering and the content-based recommendation models. As the future work, for several other data sets better results may be obtained.

REFERENCES

[1] Berners-Lee, T., Cailliau, R., & Groff, J. (1992). The world-wide web. *Computer Networks and ISDN Systems, 25*(4-5), 454-459.

[2] Jannach, D. (2011). *Recommender systems: an introduction*. New York: Cambridge University Press.

[3] Ricci, F.; Rokach, L.; Shapira, B. & Kantor, P. B., ed. (2011). *Recommender Systems Handbook* , Springer US .

[4] Resnick, P., & Varian, H. R. (1997). Recommender systems. *Communications of the ACM, 40*(3), 56-58.

[5] CNN.URL:http://money.cnn.com/magazines/fortune/fortune_archive/2006/11/27/8394347/, last visited on January 2013

[6] Adomavicius, G., & Kwon, Y. (2007). New recommendation techniques for multicriteria rating systems. *Intelligent Systems, 22*(3), 48-55.

[7] Cosley, D., Lam, S. K., Albert, I., Konstan, J. A., & Riedl, J. (2003). Is seeing believing? How recommender system interfaces affect users' opinions. *CHI '03: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* Ft. Lauderdale, Florida, USA. pp. 585-592.

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

[9] Schafer, J. B., Frankowski, D., Herlocker, J., & Sen, S. (2007). Collaborative filtering recommender systems. In P. Brusilovsky, A. Kobsa & W. Nejdl (Eds.), *The adaptive web.* pp. 291-324. Berlin, Heidelberg: Springer Berlin Heidelberg.

[10] Nichols, D. M. (1998). Implicit rating and filtering. *In Proceedings of the Fifth DELOS Workshop on Filtering and Collaborative Filtering,* Budapest. pp. 31-36.

[11] Brunato, M., & Battiti, R. (2003). A location-dependent recommender system for the web. *In MobEA Workshop,* Maggio.

[12] Osmanli, O. N., & Toroslu, I. H. (2011). Using tag similarity in SVD-based recommendation systems. *5th International Conference on Application of Information and Communication Technologies (AICT),* Baku. pp. 1-4.

[13] Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence,* Madison, Wisconsin. pp. 43-52.

[14] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. *10th International Conference on World Wide Web,* Hong Kong. pp. 285-295.

[15] Walia, R. R. (2008). Collaborative filtering: A comparison of graph-based semi-supervised learning methods and memory-based methods. *4th Annual International Conference on Computing and ICT Research,* pp. 70-84.

[16] Herlocker, J. L., Konstan, J. A., Borchers, A., & Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* Berkeley, California, USA. pp. 230-237.

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

[18] Sattari, M., Toroslu, I. H., Senkul, P., Manguoglu, M., Symeonidis, P., & Manolopoulos, Y. (2012). Geo-activity recommendations by using improved feature combination. *Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp '12),* Pittsburgh, Pennsylvania. pp. 996-1003.

[19] Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer, 42*(8), 30-37.

[20] Adomavicius, G., & Zhang, J. (2012). Impact of data characteristics on recommender systems performance. *ACM Transactions on Management Information Systems (TMIS), 3*(1), 1-17.

[21] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science, 41*(6), 391-407.

[22] Ozturk, G., & Cicekli, N. K. (2011). A hybrid video recommendation system using a graph-based algorithm. *Proceedings of the 24th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems Conference on Modern Approaches in Applied Intelligence (IEA/AIE'11),* Syracuse, NY. pp. 406-415.

[23] Symeonidis, P., Papadimitriou, A., Manolopoulos, Y., Senkul, P., Toroslu, I. H., & Toroslu, I. H. (2011). Geo-social recommendations based on incremental tensor reduction and local path traversal. *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Location-Based Social Networks (LBSN '11),* Chicago, Illinois. pp. 89-96.

[24] Tatli, I., & Birturk, A. (2011). A tag-based hybrid music recommendation system using semantic relations and multi-domain information. *Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops (ICDMW '11),* Washington, DC, USA. pp. 548-554.

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

[26] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2002). Incremental singular value decomposition algorithms for highly scalable recommender systems. *Proceeding of the 5th International Conference on Computer and Information Science,* pp. 27-28.

[27] Canny, J. (2002). Collaborative filtering with privacy via factor analysis. *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Tampere, Finland. pp. 238-245.

[28] Lemire, D., & Maclachlan, A. (2005). Slope one predictors for online rating-based collaborative filtering. *Proceedings of the 5th SIAM International Conference on Data Mining (SDM '05),* Newport Beach, CA. pp. 471-480.

[29] Das, A. S., Datar, M., Garg, A., & Rajaram, S. (2007). Google news personalization: Scalable online collaborative filtering. *Proceedings of the 16th International Conference on World Wide Web (WWW '07),* Banff, Alberta, Canada. pp. 271-280.

[30] Wikipedia. URL: http://en.wikipedia.org/wiki/Google_News, last visited on January 2013

[31] Shani, G., Brafman, R. I., & Heckerman, D. (2002). An MDP-based recommender system. *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence,* pp. 453-460.

[32] Pennock, D. M., Horvitz, E., Lawrence, S., & Giles, C. L. (2000). Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach. *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence,* San Francisco. pp. 473-480.

[33] Gauch, S., Speretta, M., Chandramouli, A., & Micarelli, A. (2007). User profiles for personalized information access. In P. Brusilovsky, A. Kobsa & W. Nejdl (Eds.), *The adaptive web* (pp. 54-89). Berlin, Heidelberg: Springer-Verlag.

[34] Balabanovic, M., & Shoham, Y. (1997). Fab: Content-based, collaborative recommendation. *Communications of the ACM, 40*(3), 66-72.

[35] Mooney, R. J., & Roy, L. (2000). Content-based book recommending using learning for text categorization. *Proceedings of the Fifth ACM Conference on Digital Libraries (DL '00)*, San Antonio, Texas, USA. pp. 195-204.

[36] Pazzani, M., & Billsus, D. (1997). Learning and revising user profiles: The identification of Interesting web sites. *Machine Learning, 27*(3), 313-331.

[37] Shardanand, U., & Maes, P. (1995). Social information filtering: Algorithms for automating "word of mouth". *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '95)*, Denver, Colorado, USA. pp. 210-217.

[38] Burke, R. (2000). Knowledge-based recommender systems. *Encyclopedia of Library and Information Science, 69*(32).

[39] Felfernig, A., & Burke, R. (2008). Constraint-based recommender systems: Technologies and research issues. *Proceedings of the 10th International Conference on Electronic Commerce (ICEC '08),* Innsbruck, Austria. pp. 1-10.

[40] Felfernig, A., Isak, K., Szabo, K., & Zachar, P. (2007). The VITA financial services sales support environment. *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI '07),* Vancouver, British Columbia. pp. 1692-1699.

[41] Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction, 12*(4), 331-370.

[42] Basu, C., Hirsh, H., & Cohen, W. (1998). Recommendation as classification: Using social and content-based information in recommendation. *Proceedings of the Fifteenth national/tenth Conference on Artificial intelligence/Innovative Applications of Artificial Intelligence (AAAI '98/IAAI '98),* Madison, Wisconsin, USA. pp. 714-720.

[43] Zanker, M., & Jessenitschnig, M. (2009). Collaborative feature-combination recommender exploiting explicit and implicit user feedback. *Proceedings of the 2009 IEEE Conference on Commerce and Enterprise Computing (CEC '09),* pp. 49-56.

[44] Melville, P., Mooney, R. J., & Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendations. *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI-02),* Edmonton, Alberta, Canada. pp. 187-192.

[45] Zheng, V. W., Zheng, Y., Xie, X., & Yang, Q. (2010). Collaborative location and activity recommendations with GPS history data. *Proceedings of the 19th International Conference on World Wide Web (WWW '10),* Raleigh, North Carolina, USA. pp. 1029-1038.

[46] Zheng, V. W., Cao, B., Zheng, Y., Xie, X., & Yang, Q. (2010). Collaborative filtering meets mobile recommendation: A user-centered approach. *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2010),* Atlanta, Georgia, USA. pp. 236-241.

[47] Zheng, V. W., Zheng, Y., Xie, X., & Yang, Q. (2012). Towards mobile intelligence: Learning from GPS history data for collaborative recommendation. *Artificial Intelligence Journal (AIJ), 184-185*, 17-37.

[48] Huang, Q., & Liu, Y. (2009). On geo-social network services. *Proceedings of the 17th International Conference on Geoinformatics,* Fairfax, VA. pp. 1-6.

[49] Lax, P. D. (2007). *Linear algebra and its applications* (2nd ed.) Wiley.

[50] Nocedal, J., & Wright, S. (2006). *Numerical optimization* (2nd ed.) Springer.

[51] Singh, A. P., & Gordon, G. J. (2008). Relational learning via collective matrix factorization. *Proceedings of the 14th ACM SIGKDD International Conference on*

Knowledge Discovery and Data Mining (KDD '08), Las Vegas, Nevada, USA. pp. 650-658.

[52] Spiegel, S., Kunegis, J., & Li, F. (2009). Hydra: A hybrid recommender system [cross-linked rating and content information]. *Proceedings of the 1st ACM International Workshop on Complex Networks Meet Information; Knowledge Management (CNIKM '09),* Hong Kong, China. pp. 75-80.

[53] Burke, R. (2007). Hybrid web recommender systems. In P. Brusilovsky, A. Kobsa & W. Nejdl (Eds.), *The adaptive web* (pp. 377-408). Berlin, Heidelberg: Springer Berlin Heidelberg.

[54] Microsoft Research. URL:http://research.microsoft.com/pubs/143146 /aaai10. uclaf.data.zip, last visited on January 2013

[55] Aristotle University of Thessaloniki, Greece. URL: http://delab.csd.auth.gr/ geosocial2/index2.html, last visited on Jan. 2013

[56] Panagiotis Symeonidis, URL: http://delab.csd.auth.gr/~symeon/index.php, last visited on January 2013

[57] Sarwar, B. M., Karypis, G., Konstan, J. A., & Riedl, J. T. (2000). Application of dimensionality reduction in recommender system - A case study. *ACM WebKDD Workshop*.

[58] MIT,URL:http://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm, [Jan. 22, 2013]

[59] WolframAlphaLLC,URL:http://www.wolframalpha.com/input/?i=frobenius+norm, last visited on January 2013

[60] Grcar, M., Mladenic, D., Fortuna, B., & Grobelnik, M. (2006). Data sparsity issues in the collaborative filtering framework. *Proceedings of the 7th International Conference on Knowledge Discovery on the Web: Advances in Web Mining and Web Usage Analysis (WebKDD'05),* Chicago, IL. pp. 58-76.

[61] Symeonidis, P., Nanopoulos, A., & Manolopoulos, Y. (2010). A unified framework for providing recommendations in social tagging systems based on ternary semantic analysis. *IEEE Trans.on Knowl.and Data Eng., 22*(2), 179-192.

[62] http://edbardsley.org/classes/15-211/lab4/lossy.html, last visited on January 2013

[63] Wikipedia, http://en.wikipedia.org/wiki/Hadamard_product_%28matrices%29, last visited on January 2013

[64] Lathauwer, L. D., Moor, B. D., & Vandewalle, J. (2000). A multilinear singular value decomposition. *SIAM J. Matrix Analysis and Applications, 21*(4), 1253-1278.

[65] Anand, S. S., & Mobasher, B. (2007). Contextual recommendation. In B. Berendt, A. Hotho, D. Mladenic & G. Semeraro (Eds.), *From web to social web: Discovering and*

deploying user and content profiles (pp. 142-160). Berlin, Heidelberg: Springer-Verlag.

[66] Brett W. Bader, Tamara G. Kolda and others. *MATLAB Tensor Toolbox Version 2.5*, URL: http://www.sandia.gov/~tgkolda/TensorToolbox/, last visited on January 2013

[67] Kolda, T. G., & Sun, J. (2008). Scalable tensor decompositions for multi-aspect data mining. *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008)*, pp. 363-372.

[68] Marinho, L. B., Hotho, A., Jaschke, R., Rendle, S., & Symeonidis, P. (2012). *Recommender systems for social tagging systems* Springer.

[69] Foltz, P. W. (1990). Using latent semantic indexing for information filtering. *Proceedings of the ACM SIGOIS and IEEE CS TC-OA Conference on Office Information Systems*, Cambridge, Massachusetts, USA. pp. 40-47.

[70] Kautz, H., Selman, B., & Shah, M. (1997). Referral web: Combining social networks and collaborative filtering. *Communications of the ACM, 40*(3), 63-65.

[71] MIT,URL:http://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm, last visited on January 2013

[72] MATLAB. (2009). *Version 7.8.0 (R2009a)*. Natick, Massachusetts: The MathWorks Inc.

[73] http://edbardsley.org/classes/15-211/lab4/lossy.html, last visited on January 2013

[74] Princeton University "About WordNet." WordNet. Princeton University. 2010. URL: http://wordnet.princeton.edu, last visited on January 2013

[75] J. Han, and M. Kamber, "Data Mining: Concepts and Techniques", Second edition, 2006, Morgan Kaufmann, USA.

[76] Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Trans.Inf.Syst., 22*(1), 5-53.

[77] Herlocker, J. L., Konstan, J. A., Borchers, A., & Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '99)*, Berkeley, California, USA. pp. 230-237.

[78] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). GroupLens: An open architecture for collaborative filtering of netnews. *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW '94)*, Chapel Hill, North Carolina, USA. pp. 175-186.

[79] Woerndl, W., Schueller, C., & Wojtech, R. (2007). A hybrid recommender system for context-aware recommendations of mobile applications. *Proceedings of the 2007*

IEEE 23rd International Conference on Data Engineering Workshop (ICDEW '07), Washington, DC, USA. pp. 871-878.

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

[81] Fan, J. and Li, R. (2003). *Local Modeling: Density estimation and Nonparametric Regression*. In *Advanced Medical Statistics*, (J. Fang and Y. Lu, eds.), 885-930, World Scientific.

[82] Hirst, G., St-Onge, D. 1998. Lexical chains as representations of context for the detection and correction of malapropisms. In *Fellbaum 1998*, pp. 305–332.

[83] Jay J. Jiang and David W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of International Conference on Research in Computational Linguistics*, Taiwan.

[84] Leacock, C., Chodorow, M. 1998. Combining local context and WordNet similarity for word sense identification. In *Fellbaum 1998*, pp. 265–283.

[85] Lin, D. 1998. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning* (ICML'98). Madison, WI.

[86] Resnik, P. 1995. Using information content to evaluate semantic similarity. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 448–453, Montreal.

[87] Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society. Series B (Methodological)*, *36*(2), 111-147.