User Behaviour Modelling in a Multi-Dimensional Environment for Personalization and Recommendation

In Fulfilment of the Requirements

for

Doctor of Philosophy

Rakesh Rawat - n5552273
(r.rawat@qut.edu.au)

Principal Supervisor: Dr. Richi Nayak

Associate Supervisor: Proff. Yuefeng Li

Faculty of Science & Technology
Queensland University of Technology
QLD 4001, Australia
December 9th, 2010

Keywords

Server Log Data Analysis, Vector Space Models, Matrix Methods of Data Analysis, Tensor Space Modelling of Web Users, Clustering, Association Rule Mining, User Profile, Group Profile, Object Profiling, Recommendation.

Abstract

Handling information overload online, from the user's point of view is a big challenge, especially when the number of websites is growing rapidly due to growth in e-commerce and other related activities. Personalization based on user needs is the key to solving the problem of information overload. Personalization methods help in identifying relevant information, which may be liked by a user. User profile and object profile are the important elements of a personalization system. When creating user and object profiles, most of the existing methods adopt two-dimensional similarity methods based on vector or matrix models in order to find inter-user and inter-object similarity. Moreover, for recommending similar objects to users, personalization systems use the users-users, items-items and users-items similarity measures. In most cases similarity measures such as Euclidian, Manhattan, cosine and many others based on vector or matrix methods are used to find the similarities. Web logs are high-dimensional datasets, consisting of multiple users, multiple searches with many attributes to each. Two-dimensional data analysis methods may often overlook latent relationships that may exist between users and items.

In contrast to other studies, this thesis utilises tensors, the high-dimensional data models, to build user and object profiles and to find the inter-relationships between users-users and users-items. To create an improved personalized Web system, this thesis proposes to build three types of profiles: individual user, group users and object profiles utilising decomposition factors of tensor data models. A hybrid recommendation approach utilising group profiles (forming the basis of a collaborative filtering method) and object profiles (forming the basis of a content-based method) in conjunction with individual user profiles (forming the basis of a model based approach) is proposed for making effective recommendations.

A tensor-based clustering method is proposed that utilises the outcomes of popular tensor decomposition techniques such as PARAFAC, Tucker and HOSVD to group similar instances. An individual user profile, showing the user's highest interest, is represented by the *top* Υ dimension values, extracted from the component matrix obtained after tensor decomposition. A group profile, showing similar users and their highest interest, is built by clustering similar users based on tensor decomposed values. A group profile is represented by the *top* Υ association rules (containing various unique object combinations) that are derived from the searches made by the

users of the cluster. An object profile is created to represent similar objects clustered on the basis of their similarity of features. Depending on the category of a user (known, anonymous or frequent visitor to the website), any of the profiles or their combinations is used for making personalized recommendations. A ranking algorithm is also proposed that utilizes the personalized information to order and rank the recommendations.

The proposed methodology is evaluated on data collected from a real life car website. Empirical analysis confirms the effectiveness of recommendations made by the proposed approach over other collaborative filtering and content-based recommendation approaches based on two-dimensional data analysis methods.

Chapter 1. Introduction

1.1 Background 1

1.2 Research Gap 3

1.3 Research Questions 9

1.4 Research Objectives 10

1.5 Research Methodology and Design 12

1.6 Structure of the Thesis 15

Chapter 2. Literature Review

2.1 Web Personalization 19

2.2. User Profile: Component of a WPS 26

2.2.1 Sources of Web Users Usage and Behaviour Information 27

2.2.2 Methodologies Used for User Profiling 30

2.2.2.1 Applicability of Clustering Methods in User Profiling 32

2.2.2.2 Applicability of Association Rule Mining in User Profiling 34

2.2.2.3 Two-Dimensional Vector and Matrix-based Methods 36

2.2.2.4 Multi-Dimensional Methods 40

2.2.3 Summary: User Profile 54

2.3 Recommender System 55

2.3.1 Recommendation Methods in Context of World Wide Web 56

2.3.1.1 Applicability of Clustering Methods in Recommendation 63

2.3.1.2 Applicability of Association Rule Mining in Recommendation . 63

2.3.2 Summary: Recommender Systems 64

2.4 Ranking Methodology 67

2.4.1 Ranking Web Search and Recommended Results 67

2.4.2 Ranking Methods in Context of World Wide Web 68

2.4.3 Summary-Ranking Methods 71

2.5 Research Gap-Web Personalization Systems 71

2.6 Summary-Literature Review 73

Chapter 3. The Proposed Research Methodology

3.1 User Profiling 76

3.1.1 Model Creation and Decomposition 80

3.1.2 Individual User Model Creation and Decomposition 81

3.1.3 Group User Model Creation and Decomposition 85

3.1.4 Object Model Creation and Decomposition ..... ……. ….88
3.1.5 Discussion: Model Creation and Decomposition 92
3.2 Clustering of Users and Objects..... 94
3.2.1 VSM Clustering of Web Users based on Search Logs 96
3.2.1.1 VSM Clustering of Web Objects based on Features 97
3.2.1.2 Limitations with VSM Clustering 98
3.2.2 Proposed Fibonacci based Clustering (FIBCLUS) 99
3.2.2.1 Necessity for Clustering Methods with the Ability to Cluster Mix Attributes Datasets..... 100
3.2.2.2 Clustering Objects based on the Proposed FIBCLUS Method 105
3.2.2.3 FIBCLUS Clustering of Objects based on Features 111
3.2.2.4 Discussion: Clustering with FIBCLUS 112
3.2.3 Tensor Clustering of Web Users based on Search Logs 113
3.2.3.1 Tensor Clustering of Web Users based on the Proposed DIF Method 113
3.2.3.2 Tensor Clustering of Web Objects Based on Features 116
3.2.3.3 Limitations with Tensor Clustering 117
3.3 Profile Creation Methods 118
3.3.1 Individual User Profile Creation Method 120
3.3.2 Group Profile Creation Method 122
3.3.3 Object profile Creation Method 127
3.3.4 Discussion: User Profiling Methods 129
3.4 Recommendations based on User, Group and Object Profiles 131
3.4.1 Recommendation based on Individual User Profile 132
3.4.2 Recommendation based on Group Profiles..... 135
3.4.3 Recommendation based on Object Profiles..... 137
3.4.4 Summary: Recommendation Methodology..... 139
3.5 Ranking Methodology for Web Search and Recommendations 140
3.5.1 Need for Ranking Recommendation Results 140
3.5.2 The Proposed Ranking Methodology for Returning Personalized Search Results 143
3.5.3 Discussion: Ranking Methods 150
3.6 Summary:The Proposed Personalization Model 151

Chapter 4. Case Study of a Sample Website

4.1 Introduction 153

4.1.1 Preliminarily Analysis 155

4.1.2 Data Filtering and Modelling 157

4.1.3 Summary: Case Study of the Sample Website 160

Chapter 5. The Proposed Clustering Methods: Empirical Analysis

5.1 Datasets Used for Measuring Clustering Methods 161

5.1.1 Web search datasets 162

5.1.2 General Datasets 165

5.2 Evaluation Criteria 165

5.3 Clustering based on Tensor Methods 169

5.3.1 Experimental Design 169

5.3.2 Evaluation Metrics 173

5.3.3 Results 176

5.3.4 Discussion 182

5.4 Evaluation of Tensor Clustering on Object Data with Features 184

5.4.1 Experimental Design of Object Data with Features 184

5.4.2 Evaluation 185

5.4.3 Results 186

5.4.4 Discussion 187

5.5 Evaluation of FIBCLUS Clustering 187

5.5.1 Experimental Design for Test Datasets 188

5.5.2 Evaluation 189

5.5.3 Results 189

5.5.4 Discussion 192

5.6 Summary: Evaluation of Clustering Methods 196

Chapter 6. Recommendation Methods: Empirical Analysis

6.1 Recommendations based on Individual Profile 198

6.1.1 Experimental Design 199

6.1.2 Evaluation Methods 200

6.1.3 Results 201

6.1.4 Discussion 205

6.2 Recommendations based on Group Profile 205

6.2.1 Experimental Design 206

6.2.2 Evaluation Methods 206
6.2.3 Results 207
6.2.4 Discussion 210
6.3 Comparison of Group and Individual Profile based Recommendation ... 215
6.3.1 Experimental Design 215
6.3.2 Evaluation Methods 216
6.3.3 Results 216
6.3.4 Discussion 219
6.4 Recommendation based on Object Profile 220
6.4.1 Experimental Design 220
6.4.2 Evaluation Methods 220
6.4.3 Results 221
6.4.4 Dicussion: Recommendation Methods 222
6.5 Summary: Recommendation Methods. 222
Chapter 7. The Proposed Ranking Method: Empirical Analysis
7.1 Evaluation of Results on Parameterized Web Search 224
7.1.1 Experimental Design: Parameterized Web Search Ranking..... 225
7.1.2 Evaluation: Parameterized Web Search Ranking 226
7.1.3 Results and Discussion: Parameterized Web Search Ranking..... 226
7.2 Evaluation of Personalized Web Search 229
7.2.1 Experimental Design: Ranking Personalized Search 229
7.2.2 Evaluation: Personalized Search 229
7.2.3 Results and Discussion: Personalized Search..... 230
7.3 Summary: Ranking Methods 232
Chapter 8. Conclusion and Future Works
8.1 Contributions 235
8.2 Research Findings..... 236
8.3 Possible Future Work 239

Bibliography 240

List of Notations:

CANDECOMP- Canonical Decomposition.

CF- Collaborative Filtering.

CLF-Common Log Format.

DEDICOM- Decomposition into Directional Components Model.

DIF-Dimensional Influence Factor.

DOM- Document Object Model.

EM- Expectation-Maximization.

ELF -Extended Log Format .

FIBCLUS- Fibonacci based Clustering.

FIT- Feature Importance based Ranking Technique.

FN-False Negative.

FP-False Positive.

F-Score- F-Measure, which is used to measure accuracy of a test using *Precision* and *Recall* values.

HOSVD- Higher Order Singular Value Decomposition.

HTML-Hyper Text Mark-up Language.

ICA- Independent Component Analysis.

IDF-Inverse Document Frequency.

IIS-Internet Information Services.

IR- Information Retrieval.

KM-k -Means.

KNN-k Nearest Neighbour.

MDD- Multi Dimensional Data.

NLP-Natural Language Processing.

NNMF- Non Negative Matrix Factorization.

OLAP- Online Analytical Processing.

Para/PARA- PARAFAC Décomposition.

PCA-Principal Component Analysis.

PLSA -Probabilistic Latent Semantic Analysis.

PLS- Probabilistic Latent Semantic.

QF-Query Frequency.

QFIDF- Query Frequency Inverse Document Frequency.

RDF-Resource Description Framework.

RSS- Really Simple Syndication.

SDD- Semi Discrete Decomposition.

SSE-Sum of Squared Error.

SVD- Singular Value Decomposition.

SVM-Support Vector Machine.

TF-Term Frequency.

TF-IDF-Term Frequency Inverse Document Frequency.

TN-True Negative.

TP-True Positive.

TSM- Tensor Space Model.

Tuck-/TUCKER- Tucker Decomposition.

URL- Universal Resource Locator.

VSM-Vector Space Model.

W3C- World Wide Web Consortium.

XM-Extended K-means.

XML-Extensible Mark-up Language.

List of Figures:

Chapter 1.
Figure 1.1 Survey of users with online spending ($) visit recommended pages [76].
Figure 1.2 Importance of Recommended items at respective website sections [76].
Figure 1.3 Recommendations disliked by percent of users [76].
Figure 1.4 Reasons for Poor Recommendations Given by Users [76].
Figure 1.5 Proposed Personalized Recommender System based on Various Profiles.
Figure 1.6 Phase Wise Details for the Proposed Personalization Model.
Chapter 2.
Figure 2.1 Flow chart of literature review methods.
Figure 2.2 Core components of a Web Personalized System.
Figure 2.3 Visualization of a matrix SVD.
Figure 2.4 Visualization of a matrix NNMF.
Figure 2.5 A Visualization of 3rd order tensor.
Figure 2.6 A Visualization of 4th order tensor.
Figure 2.7 A three-dimensional diagonal tensor with ones along the super diagonal.
Figure 2.8 Matrix unfolding of a tensor with vertical, horizontal and lateral slices.
Figure 2.9 View of tensor with various component matrices.
Figure 2.10 An example of a tensor flattened as a single matrix.
Figure 2.11 Tucker decomposition of a 3rd order tensor results in component matrices as shown.
Figure 2.12 PARAFAC decomposition of 3rd order tensor gives component matrices as shown.
Figure 2.13 HOSVD decomposition of 3rd order tensor gives component matrices as shown.
Chapter 3.
Figure 3.1 Proposed research methodology, phase-wise in detail.
Figure 3.2 Complete detailed methodology adopted for making various types of profiles.
Figure 3.3 Various User Profiles identified.
Figure 3.4 Visualization of a user tensor or an object (car) tensor in six dimensions.
Figure 3.5 Algorithm for constructing Individual Users TSM from Web log data.
Figure 3.6 PARAFAC Decomposed tensor of Individual user's searches, gives these

component matrices.

Figure 3.7 Algorithm for constructing Group users TSM from Web log data.

Figure 3.8 PARAFAC Decomposed tensor of group users searches, gives component matrices as shown.

Figure 3.9 Complete Methodology for Building Object Profiles.

Figure 3.10 PARAFAC Decomposed tensor of Object Model, gives component matrices as shown.

Figure 3.11 Algorithm for Constructing Vector Model from Web Log Data.

Figure 3.12 Tensor Representation of Patients/Disease/Symptoms Data.

Figure 3.13 Agglomerative (4 Clusters).

Figure 3.14 FIBCLUS with Agglomerative (4 Clusters).

Figure 3.15 Agglomerative (8 Clusters).

Figure 3.16 FIBCLUS with Agglomerative (8 Clusters).

Figure 3.17 Complete FIBCLUS Algorithm.

Figure 3.18 Complete User, Group and Object Profile Construction Model.

Figure 3.19 Algorithm for finding associations from a group of users.

Figure 3.20 Sample of associations found out for a cluster.

Figure 3.21 A sample user profile in RDF for a demo website.

Figure 3.22 Recommendation algorithm for a user based on his profile.

Figure 3.23 Complete Recommendation algorithm for a group of users.

Figure 3.24 Ideal Ranking plot.

Figure 3.25 Ranking method of textual and Boolean features.

Figure 3.26 Ranking method of numeric feature.

Figure 3.27 Feature Importance Technique (FIT) based Ranking Algorithm.

Chapter 4.

Figure 4.1 Prototype website structure of Web pages layout.

Figure 4.2 Seller's page details.

Figure 4.3 A snapshot of search interface for website '*Mileage Cars*'.

Chapter 5.

Figure 5.1 The Proposed Personnel Spam Tracker.

Figure 5.2 Average clustering *F-Score* of Dataset 1, where * shows the proposed methods.

Figure 5.3 Precision results of clustering on PARAFAC, DIF for Dataset 1, where *

shows proposed methods.
Figure 5.4 Recall results of clustering on PARAFAC, DIF for Dataset 1, where * shows proposed methods.
Figure 5.5 Precision results of clustering on Tucker, DIF for Dataset 1, where * shows proposed methods.
Figure 5.6 Recall results of clustering on Tucker, DIF for Dataset 1, where * shows proposed methods.
Figure 5.7 Average Inter-cluster similarity using K-means for Dataset 2.
Figure 5.8. Average **i**nter-cluster similarity results for PARAFAC decomposed tensor using EM clustering.
Figure 5.9. Average **i**nter-cluster similarity for Tucker decomposed tensor using EM clustering.
Figure 5.10. Average Inter-cluster similarity for PARAFAC decomposed tensor using K-means clustering.
Figure 5.11. Average Inter-cluster similarity for Tucker decomposed tensor using K-means clustering.
Figure 5.12. Average inter-cluster similarity using K-means for Dataset 3.
Figure 5.13 *F-Score* TSM, VSM and TSM with DIF Results of Clustering on Medical Data.
Figure 5.14 Average SSE for VSM and tensor clustering on object data.
Figure 5.15 Average inter-cluster similarity for object tensor clustering.
Figure 5.16 Average intra-cluster similarities for object tensor clustering.
Figure 5.17 Purity of EM clustering.
Figure 5.18 Purity of KM clustering.
Figure 5.19 Purity of XM (extended K-means)**.**
Figure 5.20 Purity of FIBCLUS (#1 Clustering).
Figure 5.21 Purity of FIBCLUS (#2 Clustering).
Figure 5.22 Purity of FIBCLUS(with #3 Clustering).

Chapter 6.

Figure 6.1 Profile creation data statistics.
Figure 6.2 Users searches against which recommendations are measured.
Figure 6.3 *F-Score* of all methods used for recommending to individual user.
Figure 6.4 Average *F-Score* (Top3-Top15) of all methods used for recommending to

individual user.

Figure 6.5 Average summary of *F-Score* results of matrix methods and TSM based methods.

Figure 6.6 Average *F-Score* for all top 3-15 recommendations of various methods.

Figure 6.7 Precision versus Recall curve for top 3-top 15 (T3-T15) recommendations. (Where suffix Pr=Precision and Re=Recall).

Figure 6.8. *F-Score* for the *top* ϒ recommendations cluster size wise.

Figure 6.9. *F-Score* rank-wise, *top* ϒ recommendations.

Figure 6.10 Overall average summary of recommendations results.

Chapter 7.

Figure 7.1 NDCG Comparative results of various ranking methods.

Figure 7.2. Time Comparisons between FIT and Baseline.

List of Tables:

Chapter 1.
Table 1.1 Sample dataset.
Table 1.2 Users Combined Interest vectors.
Table 1.3 Users individual interest vectors.
Chapter 2.
Table 2.1 W3C recommended log format.
Table 2.2 Some other tensor dimension reduction techniques.
Table 2.3 Some examples of applications of a 3rd order tensor.
Chapter 3.
Table 3.1 Statistically identifying user profile types from data logs.
Table 3.2 Data description for deck of cards clustering problem.
Table 3.3 Deck of cards cluster accuracy measure criteria (D1=deck1, D2=deck2).
Table 3.4 Clustering results for a single deck of cards problem.
Table 3.5 Clustering results for the two decks of cards problem.
Table 3.6. A user's top-ranked interest for cars with KMS done.
Table 3.7. A user's top-ranked interest for car models.
Table 3.8. A typical Object profile structure.
Table 3.9 Example of an object cluster C_2, where CID-=Cluster id.
Table 3.10 Example of an object profile for cluster C_k, showing top three values for four dimensions.
Table 3.11 A typical user profile structure.
Table 3.12 Table 3.12 Example of an User profile for user u_z, showing top three values for four dimensions.
Table 3.13 A snapshot of some dimensions from the *top* ϒ objects saved in a user profile.
Table 3.14 Sample Table showing how Objects clusters are saved.
Table 3.15 An example of Importance Number of Features for a car sales Website.
Table 3.16 An evaluation of Results for Query 1.
Table 3.17 Distinct score results for search query 1 using FIT.
Table 3.18 Car searched by user 1 and 2.
Table 3.19 Top 3 Result set returned for User u_1 using FIT.
Table 3.20 Top 3 Result set returned for User u_2 using FIT and with extra user profile

information.

Chapter 4

Table 4.1 Behavioural profile patterns of different categories of visitors.

Table 4.2 The top ten most popular car body types.

Table 4.3 Server log information processed into databases.

Table 4.4 Make and body type details for Dataset 1.

Table 4.5 List of number of attributes for each dataset.

Table 4.6. Number and sample of dimensions used in the tensor model

Chapter 5.

Table 5.1 Structure of processed data with relevant features, for Datasets 1-5.

Table 5.2 Processed statistics of Microsoft test searches data.

Table 5.3 Various clustering evaluation methods undertaken on different datasets.

Table 5.4 Set of keywords with spam priority.

Table 5.5 Dataset categorization.

Table 5.6. An example where all URL's about MS-Word are grouped as one.

Table 5.7 An Example where all URL's about Office OS are grouped as one.

Table 5.8 Class Distribution.

Table 5.9 Average Precision, Recall and *F-Score* for Dataset 2 using EM clustering.

Table 5.10 Average Precision, Recall and *F-Score* for Dataset 2 using KM clustering.

Table 5.11. Results *F–Score* of Microsoft Data.

Table 5.12. F-Score of clustering on E-mail spam data sets.

Table 5.13. Results of clustering purity on E-mail spam data sets.

Table 5.14. Structure of Object '*Data sets 6 and 7'*.

Table 5.15 Clustering test datasets details.

Table 5.16. Results of Purity of clustering of all datasets.

Table 5.17 Results of Entropy of clustering of all datasets.

Table 5.18 Results of *F-Score* of clustering of all datasets.

Table 5.19 Detailed Results of Clustering (Where suffix Co=Correctly Classified and In=Incorrectly Classified, EM=Expectation-Maximization, KM=K-means, XM=Extended K-means), FIB_EM , FIB_KM and FIB_XM are EM, K-means and X-Means clustering when Fibonacci altered values are taken and clustered with the respective clustering algorithms.

Table 5.20 Summary of results on test datasets.

Chapter 6.

Table 6.1 Average Precision and Recall of all methods.

Table 6.2 Comparative Average *F-Score* Results of TSM and NNMF.

Table 6.3 Average Summary of Results of TSM and NNMF.

Table 6.4 Average Summary of *F-Score* Results of Matrix methods and TSM based methods.

Table 6.5 Statistics of filtered data used for tensor modelling.

Table 6.6 Score of Top 3-15 recommendations for various methods.

Table 6.7 Average Precision, Recall and *F-Score* for CF-based methods using Euclidian similarity methods.

Table 6.8 Average Precision, Recall and *F-Score* for CF-based Methods using cosine similarity methods.

Table 6.9 Average Precision, Recall and *F-Score* for PARAFAC-1 using EM Clustering.

Table 6.10 Average Precision, Recall and *F-Score* for PARAFAC-2 using EM clustering.

Table 6.11 Average Precision, Recall and *F-Score* for PARAFAC-3 using EM clustering.

Table 6.12 Average Precision, Recall and *F-Score* for PARAFAC-1 using KM clustering.

Table 6.13 Average Precision, Recall and *F-Score* for PARAFAC-2 using KM clustering.

Table 6.14 Average Precision, Recall and *F-Score* for PARAFAC-3 using KM clustering.

Table 6.15 Average aggregate recommendations for various methods.

Table 6.16 Precision and Recall values for GPF, using PARAFC rank 1-6, (where length of associations is 2).

Table 6.17. Precision and Recall Values for GPF, using Tucker rank 1-6, (where length of associations is 2).

Table 6.18 Precision and Recall values for GPF, using HOSVD rank 1-6, (where length of associations is 2).

Table 6.19 Precision and Recall values for GPF, using PARAFC rank 1-6, (where

length of associations is 3).

Table 6.20 Precision and Recall values for GPF, using Tucker rank 1-6, (where length of associations is 3).

Table 6.21 Precision and Recall values for GPF, using HOSVD rank 1-6, (where length of associations is 3).

Table 6.22 Average *F-Score* for GPF based methods taking different association lengths.

Table 6.23 Average *F-Score* for individual user profile and group profile methods.

Chapter 7.

Table 7.1 Query Selection Criteria.

Table 7.2 A sample Profile of two users.

Table 7.3 Results of personalized search with baseline search on Dataset 5.

Statement of Original Authorship

To the best of my knowledge, the work proposed in this thesis has not been previously submitted for a degree or diploma at any other higher education institution. All work done, ideas proposed, and methods developed in this thesis are my original contribution. In the instances, where such ideas, methods, tools and techniques are used, they have been duly referenced.

Signature: **Date:**

Acknowledgements

First of all I am very grateful to my supervisor Dr. Richi Nayak for her constant support, guidance and motivation. Her pursuit of excellence motivated me to develop the new ideas discussed in this work in the form of Web data mining using tensors and methods to improve clustering. I have appreciated her helping and guiding me step by step in this long journey and enriching me with some of her vast knowledge.

I am also grateful to my associate supervisor Professor Yuefeng Li, who was always willing to assist me, no matter how big or small the issue was.

I would personally like to thank Dr. Ricky Tunny, Carol Richter, Agatha Nucifora, Nicole Levi, Mark Dwyer, Rosemary Willemse, Ann Ahlberg and everyone at QUT research support, Library services, administration and technical support including the HPC team for always being there whenever I needed them. I sincerely appreciated their zeal in resolving each issue promptly, regardless of its size.

My special thanks to all the CRC (Co-operative Resesearch Centre, Australia) team especially Dr. Julien Vayssière, Terri McLachlan, Warren Bradey and Dr. Rob Cook, who have given me constant support and shown unlimited trust in all my endeavours. I would also like to thank my research sponsors, the Fairfax Digital Media for their valuable advice, suggestions and recommendations.

I would like to personally thank each member of the ADMRG (Advanced Data Mining Research Group) Slah Alsaleh, Sangeetha Kutty, Lin Chen, Daniel Emerson, Meng Zhang, Poromin Bheganan, Paul Te Baraak, Reza Hassanzadeh, Tien Tran and Mohd Amin Afanadi for encouraging fresh ideas in our fortnightly group discussions. I am also grateful to Dr. Guy Gable and Dr. Ayon Chakraborty from the BPM (Business Process Management) group for providing valuable insights.

Last, but not the least, I would like to thank my mother (Shrimati Sampatti Devi Rawat), my family, my uncles, especially Dr. Bachan Singh Rawat and my cousins especially (Triloki Rawat, Dr. Arun Rawat and Mahabir Rawat) for showing trust in all my endeavours. I thank my wife Sunita and my sons Sarvaghya and Keanu Rawat

for their endless love, encouragement, support and patience at every stage of this research. I lovingly thank them for giving me the space, time and mental courage to achieve my goals. Apart from all these poeple thanks to everyone who were there especially Jagdeep Negi, Vijay Singh, Aiden Deem and Shubham Karodi. All of them have helped me in some way to achieve this milestone.

List of Publications: The following is the list of publications that have been published based on the work discussed in this thesis.

Refereed Conference Papers:

1. Rawat, R., Nayak, R., and Li, Y., "Clustering of Web Users Using the Tensor Decomposed Models," in Adjunct Proceedings of the 18th International Conference on User Modelling, Adaptation, and Personalization, Big Island of Hawaii, HI, USA, June 20-24, 2010, p. 37.

2. Rawat, R., Nayak, R., and Li, Y., "Individual User Behaviour Modelling for Effective Web Recommendation," in IC4E, 2nd International Conference on e-Education, e-Business, e-Management and e-Learning, Mumbai, India, IEEE, 2011.

3. Rawat, R., Nayak, R., and Li, Y., "Effective Hybrid Recommendation Combining Users-Searches Correlations Using Tensors," in 13th Asia-Pacific Web Conference Beijing, China, LNCS 6612, pp. 131—142, Springer, Heidelberg, 2011.

4. Rawat, R., Nayak, R., Li, Y., and Alsaleh, S., "Aggregate Distance Based Clustering Using Fibonacci Series-FIBCLUS," in 13th Asia-Pacific Web Conference, Beijing, China, LNCS 6612, pp. 29—40, Springer, Heidelberg, 2011.

5. Rawat, R., Nayak, R., and Li, Y., "Improving Web Database Search Incorporating Users Query Information," in International Conference on Web Intelligence, Mining and Semantics-WIMS-2011 Sogndal, Norway: ACM, 2011.

Refereed Journal Papers:

1. Rawat, R., Nayak, R., and Li, Y., "Identifying Interests of Web User for Effective Recommendations," in International Journal of Innovation Management and Technology (IJIMT), Hong Kong, Vol. 2, 1, pp.19-24, February, 2011.

Chapter 1 Introduction

1.1 Background

The wide applicability of the Web in various fields has transformed it into an unsystematic, unorganised and huge information repository. This leads to the problem of information overload where often no relevance is given to the personal interests of users [128]. When searching for information online, users in most cases have to search again, and make changes to the query raised in order to get the appropriate information [83].

The number of websites has increased nearly ten-folds in the last decade [75]. This growth has led to a flooding of information. Simultaneously with this increase in information, the number of Web users has also increased. A report [75] published online shows that in the year 2009 there were 1.73 billion Internet users worldwide**.** Another similar statistics for the current period in June 2010 shows that the number of users has increased to about 1.97 billion [186]. With the increased use of the Internet for day-to-day activities, growth of internet, e-commerce and related technologies will lead to a massive increase in the number of new websites coming in the near future. More websites will aggravate the already existing information overload problem for users, resulting in a waste of time, money and resources. Today there is so much information available on the World Wide Web that to find the relevant information as needed, a user needs some external information processing agents that can facilitate the information search.

The current information retrieval systems provide the same information to any two users. Consider the following example where two users with the same query but different preferences are provided with the same information. For example, the search for 'Cricket' by two users A and B may yield similar results. However, User A wants information about cricket, the insect, and User B, wants to know about the latest news in the sport. Lots of queries put forward by users are short and ambiguous and do not help the information retrieval systems to retrieve the desired results. As a consequence, users have to reformulate the same query many times [83]. Very often individual preferences of a user are ignored when delivering information to him. This

problem is worsened when users employ use of multiple query terms in their search. In this case, too many results are displayed, matching a few or all of the words thus, making the process of finding relevant results more cumbersome. There is a great need for personalized systems in the near future to handle this information overload problem for users. The goal of a personalized system should be to provide users with highly relevant results that match his or her information needs. This would improve a visitor's Web browsing experience. Personalization can help in saving resources, energy and time, and help in satisfying a user's information needs. In this scenario the aim of Web service providers for its users should be the approach of YGWYW or "*You Get What You Want*".

Web personalization methods try to identify the user's needs based on his past and current browsing behaviour, and predict his future information needs by recommending him information that will interest him [127], [128], [129]. Today personalization is widely adopted by many websites to retain visitors and to reduce the number of unnecessary searches made by users. Such personalization indirectly reduces the load on the server, and improves the quality of information given to users [138]. Nowadays more and more online users prefer to have personalized Web services tailored to their personnel information requirements and needs [76].

In 2006, a personalization survey conducted by ChoiceStream [76], discovered that the number of consumers willing to provide demographic information in exchange for a personalized online experience has grown dramatically over the past year, increasing 24% to a total of 57% of all responders. The survey also learned that the number of consumers willing to allow websites to track their clicks and purchases increased 34% from the previous year. The results showed no significant decline in the number of consumers concerned about the security of their personal data online, with 62% expressing concern in 2006 vs. 63% in 2005. A similar survey conducted in 2009 by ChoiceStream revealed that 74% of shoppers definitely noticed product recommendations, found them useful, and bought products based on them. High spenders eventually visited the recommended pages more often as shown in Figure 1.1. The other motivating finding was that recommendation placement does matter to shoppers as shown in Figure 1.2. In the case where recommendations are on product detail pages, the online visitor is more likely to search for it and thus is more likely to purchase it.

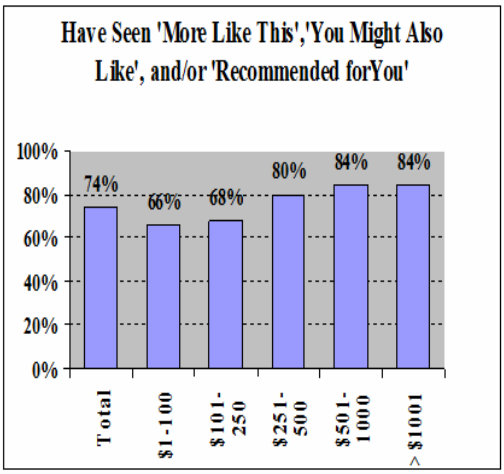

Figure 1.1 Survey of users with online spending ($) visit recommended pages [73].

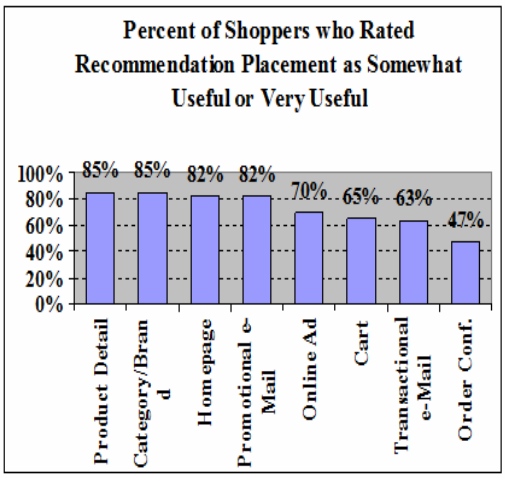

Figure 1.2 Importance of Recommended items at respective website sections [73].

1.2 Research Gap

Before beginning any discussion, two definitions of *users* and *objects* that are going to be used frequently in the relevant sections of this thesis work are discussed. A user is any visitor to the website, who does some activity like browsing the different items or products on the website. An object is any commodity that can either be a product/item or a service that has various features (or attributes) like a car, which has various features like colour, engine size, body type. In this thesis work, users and objects are considered as two different entities, where a user is a person who makes the search and an object relates to an item being searched. Therefore, the terms *'object'* and *'item'* would be used interchangeably throughout this thesis work, where both refer to the same thing.

Traditional two-dimensional data analysis methods like vectors and matrices may be good in certain data analysis methods, however, since most of the data in Web logs is MDD (Multi-Dimensional Data) , traditional vector-based methods may not be so suitable to model such data for knowledge extraction. The two most common problems faced, when two-dimensional methods are used to cluster high-dimensional data are 1) Appropriate data representation methodology 2) Knowledge extraction without loss of information and keeping latent relationships intact.

The following sample data in Table 1.1 is used to explain the representation and knowledge extraction problem with traditional two-dimensional methods.

User Id	Car Make	Car Model	Cost
1	Holden	Vectra	10,000
1	Holden	Barina	8,000
2	Ford	Falcon	15,000
2	Toyota	RAV4	10,000
2	Toyota	Corolla	15,000

Table 1.1 Sample dataset.

ID	Make 1	Make 2	Make 3	Model 1	Model 2	Model 3	Model 4	Model 5	Cost 1	Cost 2
1	2	0	0	1	1	0	0	0	1	0
2	0	1	2	0	0	1	1	1	1	2

Table 1.2 Users Combined Interest vectors.

ID	Make 1	Make 2	Make 3	Model 1	Model 2	Model 3	Model 4	Model 5	Cost 1	Cost 2
1	1	0	0	1	0	0	0	0	1	0
1	1	0	0	0	1	0	0	0	1	0
2	0	1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	0	1	0
2	0	0	1	0	0	0	0	1	0	1

Table 1.3 Users individual interest vectors.

Consider the search data organised according to what two users have searched in a car sales website (Table 1.2). For example user 1 has made two different searches associated with car make. Unfortunately, from the vector representation of this search data, it is not possible to know what make of the car was searched for what cost and which make is associated with which car model. This intra-vector information loss makes this data representation model inefficient. Another scenario when user's individual interest vectors are retained (Table 1.3) also results in some inter-vector information loss. This representation fails to inform how the different search vectors and their components (values) for the same user are inter-related to each other. They may be compared individually as vectors, but this would create extra overheads in computing, storage and processing.

The other noteworthy factor is that interest vectors would be compared using a distance measure such as Euclidean distance or cosine similarity. However, previous research [165] has shown that distance measures used for clustering or comparisons may reflect strange properties in a high-dimensional space and might not be as useful as they seem. In high-dimensional spaces, vector methods may loose valuable information, may be unable to efficiently correlate various dimensions and may be unable to magnify the latent relationships that exist between various dimensions.

Despite advances made in personalization methods, there exist some research gaps that can be filled to make a user's personal browsing a happy experience. These research gaps are:

1. Lack of efficient similarity measures for evaluating users-items or items-items similarity and lack of applicability of multi-dimensional data (MDD) modelling methods in evaluating such similarity measures: According to the Forrester Research report [180], U.S. online retail sales are projected to reach $335 billion by 2012, however, despite the steady growth, online retailers are still broadly perceived as a second choice by shoppers. Various factors such as inappropriate search results, unmatched recommendations or too many recommendations, security and trust can be ascribed to the unpopularity of online business transactions [76]. When making online recommendations a user's personal interests are often overlooked by many websites. In one survey conducted online in 2007 by Harris Interactive for popular personalization service provider website MyBuys.com, it was revealed that 62% of consumers still find that the recommendations made by most of the websites were not tailored to their own personal tastes. In another survey conducted in 2009 by ChoiceStream it was revealed that the quality of product recommendations on retailers sites declined 31% year by year with many more shoppers reporting that they received poor quality recommendations in 2009 versus 2008 (Figure 1.3).

When asked about how recommendations failed to satisfy them, online shoppers provided a variety of reasons, and it is worthy to note, the most popular reason in 2009 was that the recommendations that were given to the shoppers were unrelated to what the shoppers were searching. Recommending un-related items for a user's search is the big drawback of most of the existing recommender systems. Clearly, as can be seen in Figure 1.4, more then half of the users were recommended products that were unrelated to their search. Currently, the most popular and widely used methodologies for making recommendations include the collaborative filtering approach [73], [99], [121], [152], [161], [176]; the content-based approach [22], [104], [106]; and the hybrid approach [28], [30], [86], [114], [115], [136]. Most of these existing methods use two-dimensional data analysis techniques to find similarities between users-users, users-items or items-items [128], [172]. As clearly pointed out in the findings of these personalization surveys, recommending similar items based on such traditional

methods may not be able to map the inter-relationships between users or items to a greater extent, thus resulting in information loss, and as a result the recommendations arising out of such methods may not be liked by users.

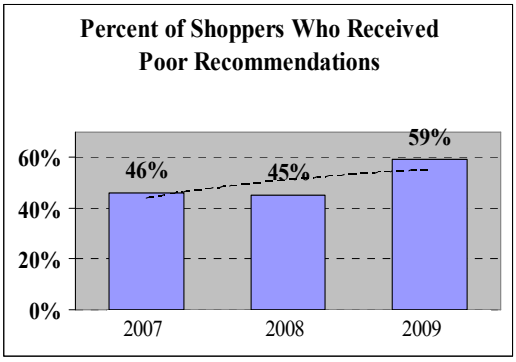

Figure 1.3 Recommendations disliked by percent of users [76].

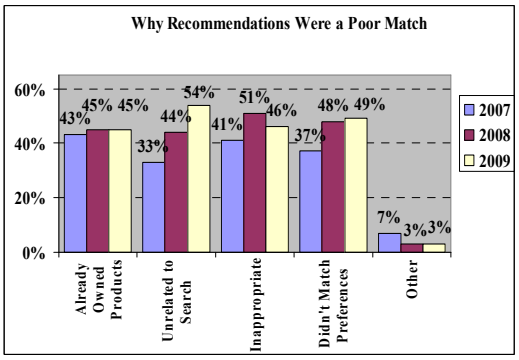

Figure 1.4 Reasons for Poor Recommendations Given by Users [76].

Web log data are complex datasets, consisting of multiple users, multiple searches with one or many objects being searched and each object having one or many parameters or properties. When making recommendations to users, each object with multiple properties is compared with other objects and based on the similarity of object properties, similar objects are recommended. Similarly, the searches made by some users are compared with similar searches made by other users. Since two-dimensional methods are mostly used to compare this multi-dimensional users-items data, many valuable relationships between users and objects are lost due to representation and inference limitations of two-dimensional methods. Apart from this, the objects being searched and users have some latent relationships that are not being fully exposed by traditional two-dimensional data models. These latent relationships may possibly be answers to questions such as what parameters are highly preferred by users, what the important attribute values are, and what the correlations between searched items are, their attributes, and the user's attributes. ? When two-dimensional methods are used for modelling user behaviour due to dimensionality limitations the correlations between users-users and users-items are not captured appropriately. Thus, there is a need to utilize MDD models such as tensors to model user behaviour and find similarity between users and items.

2. Finding users with the most similar search behaviour, in cases where searched items have multiple features: Research shows that when searching for luxury items such as cars, electronic equipment and mobile phones, users have a pre-determined state of mind [137]. Users look for the same product in different websites in the hope of finding more competitiveness and information. For example, users would only buy a make or model of a car that they had always yearned for. Their opinion is formed over a period of time and is difficult to change [3]. The search log data used in this study has also pointed out a similar finding. Of the 10,000 users studied, 54% had constantly searched for only one make and 45% had searched for a single model in the majority of their searches over a period of time. On the other hand, when buying daily consumer goods buyers can be more experimental and may often look for new brands.

Users with similar interests search for similar items or services. As an example, a person looking for a family car will have very similar searches to a person looking for the same category of car, rather than with a person looking for a sports car. The biggest challenge for any website is to identify users whose search behaviour is similar. Search behaviour comparison can be restricted to keywords (queries) or other parameters such as time, session, the page visited, all depending upon the need of the website. Accurate clustering of users can help solve the *'cold-start problem'* (which is when a first-time user visits the website). When such a user visits the website, based on his initial few searches he can be recommended items that are closest to his match. The other advantage of clustering similar users is that this information can be helpful in building group user profiles. However, looking at the MDD nature of Web log data, methods that can enhance clustering are needed.

3. Finding relevant and unique rules from searches made by similar users, in case when group user profiles are constructed: When constructing group profiles, methods such as finding associations between searches made by users [89] and finding sequential rules [118], [182] are widely used. However, in cases where the number of users in a group is small, the number of such associations and sequential rules generated is less due to inadequacy of data. On the other hand, when too many users are clustered in a group, the number of similar rules generated is very large. The behaviour of Web users is unpredictable and comparing a user's searches with identified prominent sequential patterns restricts the flexibility of a system. Let a

user u_x buys Product 'A' first and then Product 'B'. The transaction is denoted as (A, B). If user u_x is compared with other user u_y, who buys product 'B' first and then 'A', denoted as (B, A). It can be clearly seen that these methods will not be able to give good recommendations to users due to the different sequential patterns, however, the searched item sets may be same for the two users [143]. The other problem with sequential rule generation is that, in the case where a full set of sequential rules (all frequent and confident rules) are needed to be generated from frequent patterns, the process is expensive and the number of frequent patterns generated is exponential to the maximum pattern length [143]. Therefore, rules that are applicable to a group have to be mined from associations of users, rather than be mined from purely sequential patterns or frequent item sets within a group.

4. Utilization of individual user profile, group user profile and object/item profiles in synchronization to make best possible recommendations: Most model-based systems use either individual user profiles [19] to make recommendations or use item/object profiles in combination with an individual profile [10] to make recommendations. Combining group profiles (consisting of profiles of similar users) with individual and object profiles can be helpful in making unique recommendations, and can also help to improve the performance of recommender systems [135]. Most Web-based personalized systems seldom use multiple profiles to make recommendations.

5. Choosing number, type and ordering of recommendations to be made to a user: To improve the quality of recommendations made to a user, there has to be a deciding factor on the number of recommendations to be made to a user and how these recommendations should be presented (scored) for each user. Various methods [43], [84], [111], [112], [124], [194] exist which either use profile information or use the CF approach to make recommendations. They still lack the ability and flexibility to utilize all available information such as profile, CF and users current preferences. Apart from this, seldom do these methods discuss how such recommendations should be ranked or scored and presented to users. Handling categorical and mixed data types and effectively using them for scoring recommended results is another shortcoming of many ranking methods.

1.3 Research Questions

Web log data is a multi-dimensional data (MDD), and has various dimensions like searches made, time of search, browser used, geographic location etc. Most of the websites concerned with selling products and services are limited to a specific domain. There may be '*n'* numbers of items or objects that a website deals with. Each object may have multiple features based on which a user's searches are made. An example is a car website. Even though it may deal with a finite number of cars, each car has certain specifications based on which a user's searches are made. It may have commonly searched parameters such as car make, car model, body type, cost, engine size, fuel use. A few research questions that arise are:

1. Are the two-dimensional similarity measures based on vector and matrix methods, which are mostly adopted by current Web-based personalized systems [128] adequate to find best mappings between users-users or users-items?.
2. How, can Web log data be modelled, keeping all relationships between searched objects and users intact? Is it possible to find users with very similar search behaviour and objects with maximum similarity between them?

However, once this data about the user, search and session is processed, multiple searches made by different users need to be compared to find users with similar dearch behaviour. The content searched by a user can be linked with his interest. Two users with same interests may search for a near similar object. However, it is unfeasible and a waste of resources to compare each user with all other users in the database to find like-minded people having similar interests. Multiple searches made by many users need to be compared so that users with similar search interests can be clustered together. Clustering is an important part of the personalization process [128]. Identification of users with similar search interest is crucial for making effective group profiles. In cases where individual user profile creation is costly, such clustering of users and objects may be used as an alternative method for making quality recommendations to users with similar search behaviour. A few research questions that arise here are:

3. How to cluster multi-dimensional data to find similar users, when the data consists of a large number of users with each user conducting multiple searches, and each search consisting of many searched parameters?

4. Are good clustering solutions with better inter-user and inter-object similarity measures, helpful in making personalized systems that can help to improve the quality of recommendations?

Once similar users are clustered based on their searches, interesting patterns of related objects as searched by the users can be found out. One research question that arises here is:

5. How to find unique rules, within a group of users having similar interests, based on the association of the searches made by users in the group?

Object profiles or item profiles are groups of items that can be clustered together based on the similarity of features between them. Since most of the objects have clear and detailed specifications, an interesting research question here is:

6. How, can object data be clustered so that objects clustered in a group have maximum similarity between them?

People who visit a website can be classified into two categories, known or registered users and unknown or anonymous users. Known users can be identified by their server logs data, whereas it is quite difficult and time consuming to identify the anonymous users. When making recommendations to both known and unknown users some research questions are:

7. Once, when the user profiles, group profiles and object profiles are created, how to use these three profiles to mine the best possible knowledge and use the three models in synchronization, to give the best recommendation to each category of users?
8. How many recommendations should there be and how should these recommendations be ranked when presenting the results to the users?

1.4 Research Objectives

To propose an efficient Web-based personalized system, the three core components identified in this research are the user profile component (consisting of individual user model, group user and objects profile model), the recommendation component and the ranking component. Each of these three is essential in making high quality of

recommendations. Such recommendations can be made based on the information requirements and personnel preferences of each category of user, such as a known user (with profile) or an unknown user (with no profile information). To propose a personalization methodology, this research focuses on utilizing high-dimensional data modelling techniques combined with data mining techniques such as clustering and association rule mining to mine knowledge from Web log data. The major objectives of this research are:

1. Identification of efficient methods for modelling Web log data to create user profiles: Web log data consisting of multiple users, objects and the subsequent searches users have made is multi-dimensional data (MDD). This data depicts a user's actual interests. When used for finding similarity between users-users, users-items or items-items, two-dimensional data models are unable to capture latent relationships in such high-dimensional spaces, therefore loosing valuable inter dimensional relationships and information about how these dimensions interact with each other [128], [146], [172]. One of the objectives of this research is to develop methods for modelling such complex Web log data, so that there is minimum information loss across various searched dimensions, and the latent relationships between various dimensions and users can be captured more effectively.

2. Finding methods that improve clustering and thus group most similar users and objects together: Identification of similar users with similar search behaviour is important to make high quality of group user profiles. Clustering has been a popular approach to achieve this. Due to the multi-dimensionality of Web log data containing many components (searched parameters), existing clustering methods applicable on such MDD data may not be able to give optimum quality of clusters [128]. Therefore, developing methods that can enhance clustering on Web users log data is another objective of this research work.

3. Recommendation strategy for individual user, group of users based on various profiles: The three profiles created (Individual, group and object) can be used alone or can be combined to give good quality of recommendations. Individual profiles take a user's personnel interests into consideration, object profiles ensures that he is recommended similar items and group profiles can be used to make unique recommendations. Another objective of the research is how to synchronize the three categories of profiles, to make the best possible recommendations to users.

4. Ranking methodology that improves search and recommendations: A personalized recommender system delivers information to users based on their interests and needs. To achieve this, it uses the user profile information. Subsequently, a ranking function enhances the performance of a personalized recommender system by ordering recommendations as per a user's information needs. One objective of this research is to propose a ranking method which can utilize profile information and give higher preferences to a user's current interests when making recommendations.

1.5 Research Methodology and Design

This thesis proposes a multi dimensional data modelling (MDD) approach to build user profiles and subsequently make recommendations. To build user profiles the method uses the implicit information about users by using their search log data. In order to find the correlations between users and objects according to their usage of items, identify what features play an important role in a user's search and what the prominent searched parameters are, this thesis proposes to use the high-dimensional data models also known as tensor space models [96], [165]. Various methods are proposed to construct profiles such as individual user's profiles, group users profiles and object profiles.

An individual user profile finds the most relevant items as searched by a user and then helps in making recommendations. Group profile construction is a collaborative approach, which groups users based on their common searches and then finds users–items correlations within a group. Once users are clustered using the proposed tensor-based clustering method, the associations shared by a group of users (represented as top Υ items), are used for making recommendations within the group. The complete personalization model adopting different methods of recommendations is explained in Figure 1.5. When a Web user directs a query to a website, the very first objective is identification of such user. If he is a known user, then his profile is used for making recommendations. If the user is an anonymous user, then a group profile is used, where a user's interest are mapped with other people having similar interests. Object profiling consisting of similar objects in a group can also be used to give more personalized recommendations to users. These recommendation results can than be given to the ranking component whose objective is to rank recommended results based on a user's current preferences and needs.

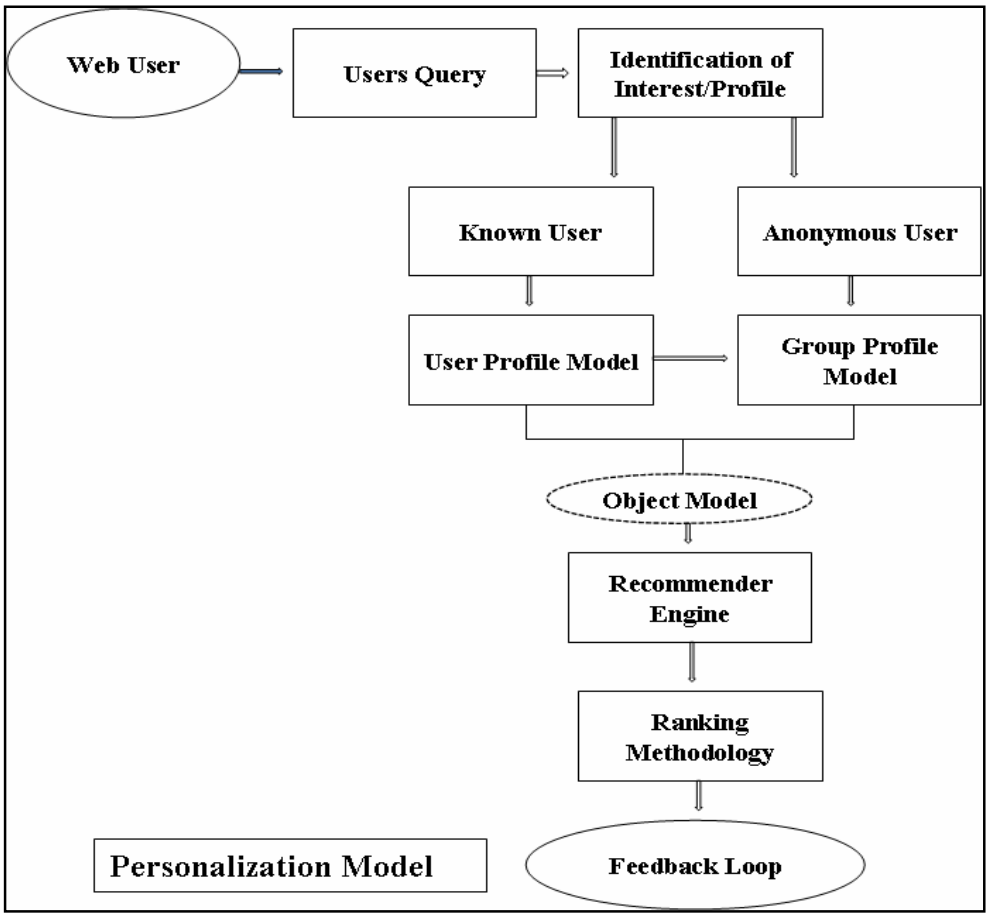

Figure 1.5 Proposed Personalized Recommender System based on Various Profiles.

In addition, to achieve the objective of constructing such personalization models (as shown in Figure 1.5), this research has been divided into six phases. These phases are shown in Figure 1.6. In the first phase all necessary data is pre-processed, filtered and arranged according to user and search. In the second phase various models such as individual, group and object are created from the datasets. Next these models are decomposed to mine interesting patterns of users and objects. In the next phase the decomposed data of users and objects is used to cluster similar users and objects. In the fourth phase individual, group and object profile models are created. Fifth phase identifies a recommendation method that is used for making recommendations to various categories of users. The sixth phase proposes a ranking method that is adopted to make the best recommendations to a user.

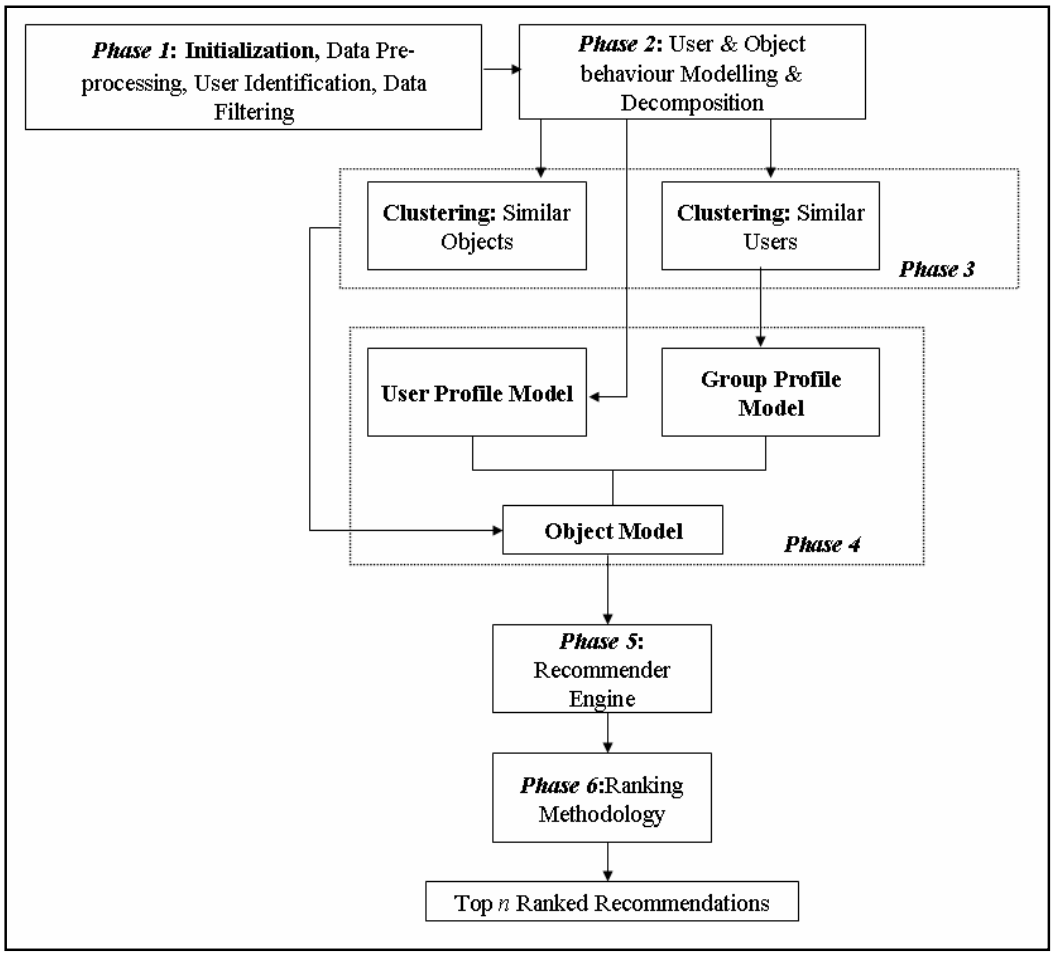

Figure 1.6 Phase Wise Details for the Proposed Personalization Model.

A brief summary of contributions made in the Thesis: The following is the list of contributions made in the thesis.

1. To find the closest similarity between users-users, user-items and items-items the thesis has proposed to use MDD techniques like tensors for modelling user and object behaviour.

2. The other contribution outlined in this thesis is introducing methods of building various users, group and object profiles.

3. For clustering similar users, this thesis proposes to do clustering on the last dimensional matrix using the proposed DIF (Dimension Influencing Factors) method. Further to improve the quality of clustering solutions obtained, the thesis proposes to cluster items/objects using the proposed FIBCLUS

similarity measure, so that inter-object distances are maximised and intra-object distances are minimized, thus enabling good clustering solutions.

4. Unlike most of the personalized systems, those use either the user model, or the CF approach, or both, this research proposes to use multiple profiles (individual, group and object profiles) in conjunction to make best possible recommendations to different categories of users.

5. This thesis proposed a ranking methodology that can recommend based on user's current search; user's individual profile information; and in case no profile information is available it can use the CF approach to give best results matching a user's interest.

1.6 Structure of Thesis

Chapter 1: It discusses the information overload problem and outlines the need for creating personalized systems for making better recommendations. A need, with a solution in the form of making hybrid systems consisting of user, group and object profiles is outlined. The other important gap that was discovered, and is a major cause of poor performance of recommender systems are the methods adopted to find correlations between users-users and user-items. For evaluating Web data, which is high-dimensional data, a multi-dimensional data modelling technique based on tensor decomposition is proposed in this thesis.

Chapter 2: In this chapter various literatures which is relevant to the research is discussed in detail. The section starts with an overview of personalization methods. It then discusses the three important components the user profile, recommendation and ranking methods of a personalization system. Next various approaches for creating user profiles are discussed. Clustering and association rule mining, which are popular data mining techniques and have been used in this research when creating user profiles, are discussed in brief. In the next section two-dimensional data modelling techniques which are used in user profiling for evaluating users-items or users-users similarity measures are discussed. Next, multi-dimensional data modelling techniques based on tensors including their properties and popular decomposition methods are

discussed. In the next section various recommendation approaches are discussed in addition to the advantages and limitations of each category of recommender systems. In the next section various ranking methods based on different similarity measures of attributes are discussed. Finally, the chapter discusses some research gaps identified in terms of Web personalized systems and concludes with a summary of the literature.

Chapter 3: Chapter 3 discusses the proposed research methodology in detail. In this chapter various methods developed for building a personalized Web system are discussed in detail.

Section 3.1: In this section, various user model creation methods such as models for individual users and group users are discussed. Further, to enhance recommendations object model creation methods, consisting of similar objects in a cluster are also discussed.

Section 3.2: This section discusses about clustering methods to cluster the different models such as models created using vectors, matrices and tensors. Clustering on the tensor models with and without the proposed DIF (Dimensional Influencing Factor) methods is also discussed. Applicability of the proposed FIBCLUS clustering method to cluster objects is also discussed.

Section 3.3: This section discusses about various profile construction methods using vector and tensors for individual user, group users and object profiles.

Section 3.4: This section discusses about recommendation methods using the various profiles such as individual user profiles, group users profiles and object profiles.

Section 3.5: This section first identifies the need to use ranking methods while delivering search results or recommendations to a user. Next the proposed ranking function FIT (Feature Importance Technique) based ranking method is described in detail.

Chapter 4: This chapter does a case study on a real car sales website which has been named as '*Mileage Cars*' in this thesis. It identifies the various profile categories, how data pre-processing is done to create various profiles and finally concludes with the creation of various tensor models for the website.

Chapter 5: This chapter discusses all the experiments related to clustering of tensor-decomposed objects, using various clustering methods such as K-means, X-means and EM. Evaluation of vector, tensor and the proposed DIF clustering on tensor models of

such Web log data is discussed. Evaluation of the proposed similarity method (FIBCLUS) for enhancing clustering is also discussed in detail.

Chapter 6: This chapter details the experiments conducted to evaluate the efficiency of recommendations methods based on individual user, group user and object profiles.

Chapter 7: This chapter details the experiments conducted to evaluate the efficiency of methods of ranking. Two evaluation methods are used. The first evaluates ranking performance by utilizing user profile information to make recommendations to a user and the second employs no personalized information to rank search results.

Chapter 8: The final chapter summarises the contributions and findings of the research and concludes with the potential for future research in the development of MDD techniques for personalization.

Chapter 2 Literature Review

This chapter discusses the various research work related to this thesis. Each section discusses briefly the techniques, reviews existing methods that are widely used for personalization and concludes with a summary of the section. Figure 2.1 gives an overview of the literature discussed in this thesis. The first section discusses about personalization and the various approaches that are commonly used for Web personalization. Since, user profiling, recommendation and ranking are all important components of a personalization system, these three are discussed separately in the next sections.

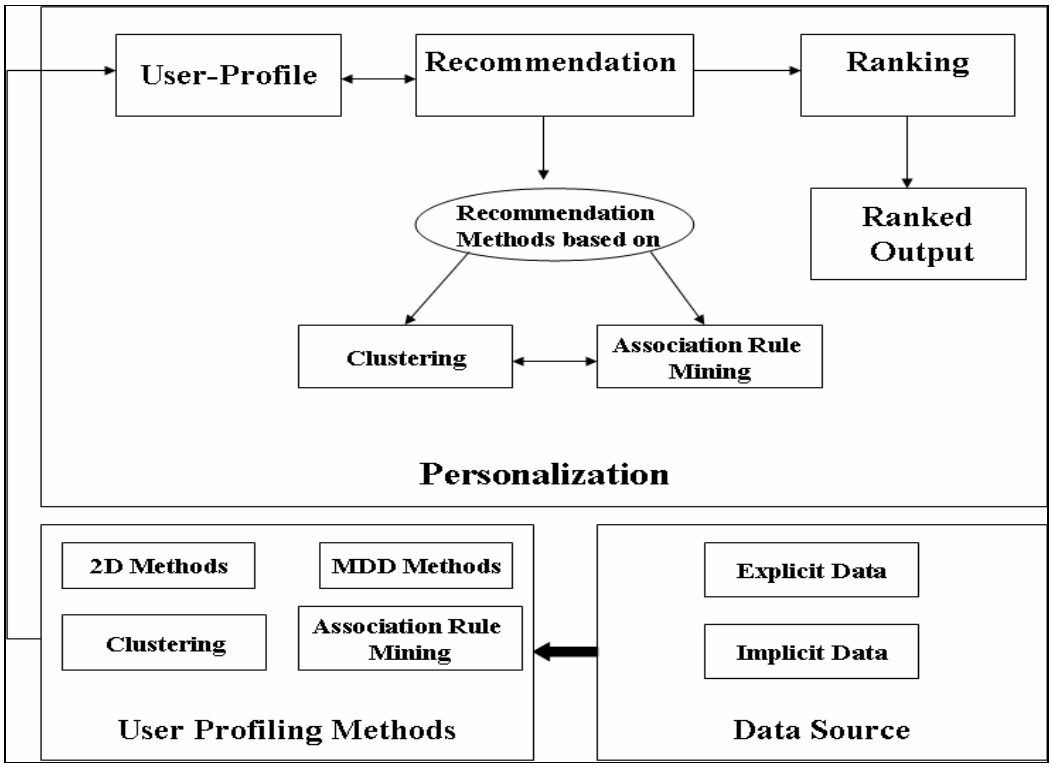

Figure 2.1 Flow chart of literature review methods.

The user profile section starts with the discussion about various data sources and formats that can be used in profile building. It then talks about various approaches used for profile building. Applicability of popular data mining techniques like clustering and association rule mining, used for building user profiles is also discussed. Next, there is discussion about the popular two-dimensional data analysis techniques used extensively to find similarity between users and objects when

building user profiles. Finally, a brief overview of tensors with details about the popular tensor decomposition techniques is given. The next Section 2.3.1, discusses about the recommendation methods, an important component of a Web personalization system. The section discusses about various methods widely used and how clustering and association rule mining have been used by recommender systems. Section 2.4 gives a brief introduction to ranking methods in terms of common Web search and ranking results in terms of recommender systems. Section 2.5 identifies some research gaps with regards to Web personalization and finally Section 2.6 gives a summary of the literature.

2.1 Web Personalization

Personalization can be described as a step closer to the semantic Web vision [123], [192]. An Intelligent Web is what semantic Web looks forward to. Due to the applicability of Web in diverse tasks like business transactions, knowledge acquisition by individual users, simply browsing for pleasure, personalization has gained momentum [64], [123], [192], [193]. There is a difference between customization and Personalization [127]. In customization, a user has to specifically outline what he wishes to see. On the other hand, personalization is a dynamic task.

In order to improve websites and retain visitors, the major objective of any personalized service is to provide better-individualized services to each category of users. Personalization methods allow Websites to know about their users. Since, this information is streamlined from the vast amount of available information it reduces unnecessary searches for a user. When a user's personnel browsing history is known it just becomes a matter of finding interesting patterns in the data. Once this has been achieved, the user can be served with data and information that matches his previous browsing behaviour [50].

The data needed for personalization can be collected either explicitly, by asking consumers to disclose information about themselves (like in registration or subscription forms), or implicitly, without the knowledge of the users, like data from the server, proxy or their Web browser history. Personalization may include content personalization (recommending content which may interest a user), or link personalization (displaying links, which a user may click) and structure/navigation personalization (reorder the website as per a user's need) [14].

Based on the methods adopted [128], there can be three types of personalization approaches 1) Rule-based 2) Content-based and 3) Collaborative filtering. A fourth category compromising a combination of these three, called as hybrid is not discussed here, as it is a combination of any of these three methods. Hybrid approach used for personalization is discussed in the recommendation Section 2.3 of this thesis.

1) *Rule-based personalization systems* are either manual, or are automatically able to generate decision rules that can be used to recommend items. Manual rule-based personalized systems allow rules to be generated based on demographic, psychographic, or other personal characteristics of users [128]. Most rule-based systems depend on explicit information which makes such systems outdated over a longer duration, since profiles created based on rules tends to be outdated [128]. User's interest change over a period and the information a user may have initially put in his profile may currently be of no interest to him. Some widely used techniques using rule-based approach for making personalization are:

a) Usage of ontology in building rule-based personalized systems: Ontology based methods in general create user profiles represented as ontologies and then compare ontologies either with other users or try to find relevant items based on users ontology. To compare or recommend these methods usually use the domain ontology. However, one major drawback of these methods is that building domain ontology is difficult, requires proper domain knowledge and takes a considerable amount of time. Some widely used methods that fall under this category are discussed below.

In this work [84], an ontology based user model, called user ontology, for providing personalized information service in the semantic Web is proposed. As pointed out by the researchers that most of the existing approaches only use concepts and taxonomic relations for user modelling, however this user ontology model utilizes concepts, taxonomic relations, and non-taxonomic relations in a given domain ontology to capture the users' interests. A set of statistical methods are provided to learn a user ontology from a given domain ontology and a spreading activation procedure for inferencing in the user ontology. The user ontology model with the spreading activation based inferencing procedure has been incorporated into a semantic search engine, called *OntoSearch*. Experimental results, based on the *ACM* digital library and the *Google* Directory highlight the improvements in personalized search.

b) Usage of machine learning methods in building rule-based personalized systems: The general idea behind these methods is to train models based on users search data and then use some machine learning or artificial intelligence techniques like neural networks to predict the user's interests.

In one such personalization research work [195], a two-way representation of a user's semantic profile firstly from a lower-level semantic representation (collecting users activities online over a time duration) and using a standard machine learning algorithm to detect user convergence, and secondly a higher-level semantic representation which detects changes in user activities are used. Once changes are detected, the higher-level semantic representation automatically updates the user profiles and reinitializes the system.

c) Usage of data mining methods in building rule-based personalized systems: Data mining methods usually model the users search data using various techniques like clustering, classification and then analyse it to find interesting patterns. These patterns are then used for making recommendations to the users. Some methods that utilize this approach are discussed briefly.

A method classified under this approach uses a graph-based approach [119], and uses a weighted projection of the user-object bipartite network to study the effects of user tastes on the mass-diffusion-based personalized recommendation algorithm. Here a user's tastes or interests are defined by the average degree of the objects he has collected. Experimental results show that when the data is sparse, the algorithm gives more recommendation weight to the objects whose degrees are close to the user's tastes, and when it is dense, it assigns more weight to the objects whose degrees are significantly different from user's tastes.

d) Usage of information retrieval (IR) methods in building rule-based personalized systems: IR based methods firstly try to identify a user's needs. To identify a user's needs various sources of information such as identifying a user's actions, asking a user his needs explicitly, or utilizing user profiles are employed. Once these needs are identified, information retrieval is done keeping the needs in highest consideration. As an example of a prominent work [27], that proposes a personalized user model based on an individual user's needs, and adopts an IR based approach based on explicit user actions. Such identification of objectives is similar to *WebWatcher* [15], where the concept of objective (also called goal) is used.

One drawback with such systems is explicitly asking users, the objective of their search prior to making any recommendations. The proposed approach is implemented in the prototyped personalized system *METIORE*, which provides access to the bibliographic references of research publications available at the library of the computer laboratory centre LORIA, France.

2) ***Content-based personalized systems*** mostly rely on past information of users searches, expressed as query or ratings of items. A user profile consisting of user's interest is mostly created by such methods. Such systems compare item features and recommend similar items matching a user interest. As pointed out by Mobasher et al in [128] these systems often rely on document modelling techniques with roots in information retrieval [155] and information filtering [20]. In most of these methods both user profiles, as well as, items are represented as weighted term vectors (e.g., based on TF-IDF term-weighting model [128] and item or user similarities are evaluated based on computation of vector similarities (e.g., using the Cosine similarity measure) or using probabilistic approaches such as Bayesian classification. One well-known drawback of content-based personalization systems is overspecialization, which is when a user may not be recommended unique items, which he had never rated. However as also pointed out by [128], [164] personalization based recommendation is helpful when users are recommended items, which are unique and may interest them. The other drawback with such systems is that since items consisting of features, and users searches consisting of many dimensions are mapped to each other, based on some two-dimensional similarity measures, such methods may not be as effective for finding item-items or users-items similarities [128]. Some widely used techniques that adopt a content-based approach are discussed below.

a) Usage of ontology in building content-based personalized systems: In one of the work [53] a semantic Web personalization system, focusing on word sense disambiguation techniques, which can be applied in order to semantically annotate the website's content. The underlying idea is to integrate usage data with content semantics, expressed in ontology terms, in order to produce semantically enhanced navigational patterns that can subsequently be used for producing valuable recommendations.

Another work [110] that proposes a new Web search personalization methodology which captures the user's interests and preferences in the form of concepts by mining search results and their click throughs. Further classification of concepts into content concepts and location concepts, by creating an *Ontology-based, Multi-Facet (OMF) model* is done, to precisely capture the user's content and location interests and hence improve the search accuracy. Lastly, based on the derived ontologies and personalization effectiveness, an SVM model is trained to adapt a personalized ranking function for re-ranking of future search. Experimental results show that *OMF* improves the *Precision* significantly when compared with baseline search methods.

Another work by [44] describes a personalized search approach involving a semantic graph-based user profile issued from ontology. User profiles are built using a score propagation that activates a set of semantically related concepts, in the same search session using a graph-based merging scheme. A session boundary recognition mechanism based on tracking changes in the dominant concepts held by the user profile relatively to a new submitted query using the *Kendall* rank correlation measure is also defined. Finally, personalization does re-ranking of the search results of related queries using the user profile.

b) Usage of machine learning methods in building content-based personalized systems: Some methods which are classified under this approach are like the one proposed by researchers [42]. In this work a feature-based machine learning approach for personalized recommendation that is capable of handling the cold-start problem is discussed. User profiles reflecting user's interests, where such profiles are updated with temporal characteristics of the content, e.g. popularity and freshness. Based on all features in user and content profiles, a predictive bilinear regression model is developed to provide accurate personalized recommendations of new items for both existing and new users. This approach has an offline component with light computational overhead compared with other recommender systems that require online re-training. Superiority of results is verified on a large-scale data set collected from the *Today-Module* on *Yahoo*.

c) Usage of data mining methods in building content-based personalized systems: In one of the works [41], the researchers employ the user's product-specific knowledge and mines his/her own desire on appropriate target products as a part of personalization process to construct the overall electronic commerce strategy for businesses. This work illustrates a novel Web usage mining approach, based on the

sequence mining technique. It takes into consideration a user's navigation behaviour, to discover patterns. A footstep graph is used to visualize the user's click-stream data so that any interesting pattern can be detected more easily and quickly. A novel sequence mining approach is used to identify pre-designated user navigation patterns automatically and integrates Back-Propagation Network (BPN) model smoothly. Empirical research showed that the proposed approach can predict and categorize the users' navigation behaviour with high accuracy.

3) ***Collaborative filtering (CF) based personalized Systems*** have been very effective in making group recommendations. A typical success example is *Amazon.com*. CF-based systems are either memory based or neighbourhood based, and make recommendations to users based on their ratings of items. A collective approach is used to identify most popular items, and recommendations are made based on a user's search matching these popular items. In spite of being a popular way of making recommendations, CF-based approach has some limitations like recommending a new item without ratings or recommending to a new user who has not rated any items. Scalability is another issue of such systems, as finding the neighbourhood of item/users is mostly an online process. To overcome these shortcomings various measures like similarity indexing [5], dimensionality reduction [158] (to reduce real-time search costs and overcome sparsity problem) and offline clustering [179] of user records (where users are recommended items from the cluster they belong) have been proposed [128]. Another CF-based approach, the model-based, is able to handle such real time recommendation issues by creating user models. In such user models, only highly ranked results matching a user's profile are recommended. Model based systems have two variations the *item-based* collaborative filtering [128], [157] and the *user-items* based CF method. In *item-based* models rating, as provided by a user to items are found out by creating item-item similarity matrix and similar items as searched by a user are recommended. All this is done as an offline process. Evaluation of the item-based CF approach [48], [128] has shown that item-based CF provides recommendation of similar quality like the memory-based CF approach. The other approach *users-items* compare all searches made by users with relative items and finds inter-relationships between such items and users. The User-to-Item correlation methods combine interests of a group of people to find the highest rated interests and

then interests consisting of items/ products/people are recommended to the individuals belonging to a group [93].

A considerable amount of work has been done in developing personalization systems, and various techniques have been used. The majority of systems and tools for Web personalization employ Web usage mining, which discovers interesting usage pattern by analysing the Web server logs. Some recent work on personalization using data mining methods and employing CF approach are discussed below.

In this work [191], a personalization model for tag-based recommendation is discussed using CF approach. The methodology also considers the personal vocabulary for making recommendations. Experiments are done on a real-world dataset the *Del.icio.us,* which demonstrates that the use of tag information can significantly improve the accuracy of personalized recommendations

In another work [151], an E-learning personalized architecture for a personalization system to facilitate Web mining is proposed. The Web mining tool (*AHA*!) is developed with an integrated recommender engine, to help the instructor to carry out the whole Web mining process. The objective is to recommend a student on his next visit, the most appropriate links/Web pages within the *AHA*! System.

A hierarchical framework for personalized movie recommendation is proposed in [117]. A CF user-item based approach using weekly ranking information of movies is utilized for movie association and recommendation. Then, an integrated graph with both movie content and user preference is constructed to generate a dynamic movie synopsis for personalized navigation.

This work [32] investigates personalized social search based on the user's social relations, where search results are re-ranked in relation to individuals in the user's social network. Comparison with topic-based personalization, that is, a user's related terms aggregated from other social applications, is considered. Results from the off-line study and user surveys show that social network based personalization significantly outperforms non-personalized social search.

Components of a Web Personalization System: The three components that are the core of any Web Personalization System (WPS) are

1. User Profile.
2. Recommender Engine.
3. Ranking Methodology or Recommender Ranker.

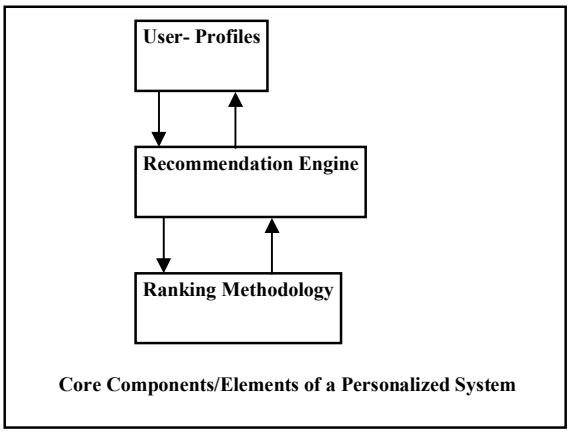

Figure 2.2 Core components of a Web Personalized System.

Each of these components works in conjunction with each other (Figure 2.2) to enhance the Web browsing experience of a user. Each of these components is discussed in detail in the following sections.

2.2 User Profile: Component of a WPS

Most Web-based personalized systems utilize user profiles to make recommendations [43], [44], [59], [61], [63], [65]. User profile is an important component of a Web personalized system. A user profile is a user's browsing behavioural model, which is constructed after analysing the past browsing behaviour of a user. The information that may be analysed is the websites he/she frequently visits, the content he sees, and time spent at each site. This information is critical in determining and predicting his futuristic Web browsing habits [3], [127]. Another objective of creating user profiles is that similar search criteria given by many users would yield the same results, regardless of individual preference. Thus, the other means to identify information, as which may be liked by a user is by analysing their user profiles [90].

Types of User Profiles: Typically, there are three types of user profiles:

1) Static (or explicit, obvious or extended) user profiles – Such profiles are built on information provided by the user in the registration or subscription forms of a website. Such information as demographic information (location, country), personal (name, age, education) and behavioural (such as preferences) may be used for creating individual or group user profiles [59].

2) Dynamic (or implicit or less obvious) user profiles – Such profiles are inferred by the click stream analysis of a user's behaviour based on their activity extracted from Web server logs or by the agent that monitors them.

3) Non-obvious user profiles – Such profiles may combine explicit as well as implicit data and may use data mining, or machine learning or IR or probabilistic approach. An example of such a profile [131] may contain the actual numbers of a user's session count and a list of weights (*w*) representing the users interest in the key topics.

To construct the profiles, information needed can be acquired from users either explicitly or implicitly or both.

2.2.1 Sources of Web Users Usage and Behaviour Information

a) Explicit Information: Such information is directly given by the users in various registrations, subscriptions, Web payment websites, or in ratings or rankings given to certain products or features. Apart from these sources explicit user comments about a product or topics are also a source of such information.

b) Implicit Information: A server log is a file (or several files) automatically created and maintained by a server for recording activities performed by users over it. A typical example is a Web server log, which maintains a history of page requests. The W3C (World Wide Web Consortium) maintains a standard format for Web server log files, but other proprietary formats exist. Information about the request, including client IP address, request date/time, page requested, HTTP code, bytes served, user agent and referrer are the typical items that are added to this file. This data can be combined into a single file or separated into distinct logs, such as an access log, error log or referrer log. However, server logs typically do not collect user-specific information [163]. These files are usually not accessible to general internet users, but only to the Webmaster or other administrative person.

A statistical analysis of the server log can reveal interesting traffic patterns over a period of time. Such knowledge can be the referrer or user agent software that is popularly used by users; website sections that are mostly viewed by users; or the product catalogues that users have browsed. Also, much more information that is similar can be mined from server logs. Thus, in brief, by doing an analysis of Web

server log data, a lot of valuable information can be mined which can help in efficient Web site administration and managing website resources. This information can also assist the website in launching promotional and marketing campaigns.

Depending on the configuration settings of the website administrator, settings of the user's personal system and settings at the proxy server level, log files can be located at three different places: 1) Web server logs, 2) Web proxy server logs, and 3) Client browser logs.

The server logs are more reliable and provide more accurate picture of the usage information of a user, however due to the privacy issues, complexity of pre-processing [19], and the sensitive and personal information such logs may contain, they are often ignored or are used rarely except in few simple analytics relating to website traffic and other site maintenance activities. Seldom are logs used widely for making user profiles or making recommendations.

The other noteworthy thing is that server logs do not record cached pages visited. The cached pages are summoned from local storage of browsers or proxy servers. The second source of log information is the proxy logs. Proxy servers act as gateways. The proxy server takes HTTP requests from client browsers and passes them to a Web server. Web pages returned by a Web server are then passed to the users thorough the proxy server.

One disadvantages of log information is that sometimes the request interception is limited, rather than covering most requests. The third category of log information is available from the client logs. Most websites employ HTTP cookies to extract such information. Cookies are pieces of information generated by a Web server and stored in the users' computers. Whenever a user visits the website in future, the user is identified by the Web server based on these cookies. However, tracking client browser logs have certain drawbacks which include security and privacy issues; deployment and compatibility issues; disabling of cookies and agent softwares that can monitor website usage. Such issues need to be addressed effectively before any personalization can be done for users.

Various Representation Formats of Users Implicit Data: The data available in Web server log consists of various formats such as Apache Common Log File Format, CERF Net, Cisco PIX, Gauntlet, IBM Firewall, IIS standard/Extended, Microsoft Proxy, NCSA Common/Combined, Netscape Flexible, Open Market Extended, Lotus Domino Log File Format and Raptor Eagle. Some popular log formats are:-

1) Common Log Format.

2) Extended Log Format.

3) Microsoft IIS Log Format.

4) W3C Extended Log File Format (IIS 6.0):

W3C Extended Log File Format (IIS 6.0): The W3C Extended log file format is the default log file format for IIS [125]. It is a customizable ASCII text-based format. The IIS Manager can be used to select which fields to include in the log file, allowing the maintaining of log files as small as possible. Field Prefixes has the following meaning, which appears on the "Appears as" column below: s-server actions, c-Client Actions, CS- Client to Server Actions, and SC- Server to Client Actions. Table 2.1 explains the fields extracted when this format is chosen by system administrators.

No	Field	Appears as	Description
1	Date	Date	The date on which the activity occurred.
2	Time	Time	The time, in coordinated universal time (UTC), at which the activity occurred.
3	Client IP address	c-imp	The IP address of the client that made the request.
4	User Name	cess-username	The name of the authenticated user who accessed your server. Anonymous users are indicated by a hyphen.
5	Service name and instance number	s-site name	The Internet service name and instance number that was running on the client.
6	Server Name	S-computer name	The name of the server on which the log files entry was generated.
7	Server IP address	s-ip	The IP address of the server on which the log file entry was generated.
8	Server Port	s-port	The server port number that is conFigured for the service.
9	Method	cs-method	The requested action, for example, a GET method.
10	URI Stem	cs-uri-stem	The target of the action, for example, Default.htm.
11	URI Query	cs-uri-query	The query, if any, that the client was trying to perform. A Universal Resource Identifier (URI) query is necessary only for dynamic pages.
12	HTTP Status	sc-status	The HTTP status code.
13	Win 32 Status	sc-win32-status	The Windows status code.
14	Bytes Sent	sc-bytes	The number of bytes that the server sent.
15	Bytes Received	cs-bytes	The number of bytes that the server received.
16	Time Taken	time-taken	The length of time that the action took, in milliseconds.
17	Protocol Version	cs-version	The protocol version —HTTP or FTP —that the client used.
18	Host	cs-host	The host header name, if any.
19	User Agent	cs(User-Agent)	The browser type that the client used.
20	Cookie	cs(Cookie)	The content of the cookie sent or received, if any.
21	Referrer	cs(Referrer)	The site that the user last visited. This site provided a link to the current site.
22	Protocol Substatus	sc-substatus	The sub-status error code.

Table 2.1 W3C recommended log format.

No matter in which format a user's log data is stored, once the user's search log data is pre-processed, and various important features as needed in the research are

identified, such features can be extracted. Subsequently various data processing techniques such as filtering, removing duplicates and arranging user's data session wise can be performed.

2.2.2 Methodologies Used for User Profiling

There are various types of user profile construction approaches. This research has classified these approaches based on five broad categories. These categories are 1) Data Mining, 2) Statistics and Network analysis, 3) Information Retrieval, 4) Machine Learning and 5) Cognitive Theory Based Approach.

Some user profiling approaches that fall under each of these categories are further described in the following section.

1) Data Mining Approach: This type of method employs data mining techniques such as frequent pattern and preference mining as in [74], [82], [92]. Frequent and preference mining is a heavily researched area in data mining. The general methodology adopted by such methods for building user profiles is clustering similar users based on their searches or page views and then finding similarities between user searches by applying data mining techniques such as association or sequential rule mining to derive interesting patterns for a user or group of users.

(2) Statistics Approach: The use of descriptive statistics to extract knowledge from Web log data has been introduced by [167], through analysing the session files. Various statistical tools such as frequency, mean, and median on variables i.e. page views, viewing time and length of a navigational path were applied to derive knowledge about the behaviour of Web users. In Addition, Web log files analysis using statistical approach as proposed by [168] allows for a broader perception of user behaviour and potential to improve user profiling. Their approach includes several methods such as statistical inference, graph analysis and profile generation.

(3) The IR Acquisition Approach: Another approach that uses the probability grammar-based approach model instead of descriptive statistics can be found in [85]. A unified framework for the discovery and analysis of Web navigational patterns based on Probabilistic Latent Semantic Analysis (PLSA) is proposed. PLSA uses the probabilistic technique of the latent semantic factor to generate a page view with a probability value. In one work, the application of a regression concept called Support Vector Regression (SVR) [87] is applied on users click stream data to find Web pages

access patterns. SVR is a regression version (a type of statistical learning models) of Support Vector Machine (SVM) that enables one to make an efficient missing value imputation model to pre-process sparse Web log.

In another work, researchers (40) have the proposed probabilistic latent semantic model (PLSA) for characterization of usage patterns and for predicting the probability of a user's visiting a particular usage segment. PLSA (Probabilistic Latent Semantic Analysis) has many applications in the field of information retrieval and filtering, text learning and many related fields. After the model has generated some usage patterns of users a recommendation algorithm is used to provide dynamic content to the user. The biggest advantage and flexibility of this model is that details can be personalized dynamically based on probabilistic outcome, even from an anonymous user's session or a user with low browsing history. In the case where more browsing data is available, personalization becomes easier and the probabilistic model shows higher level of accuracy and *Precision*. An ontology of user profiles is constructed by exploiting the user locality model.

In another work [19], a rule-based model of a user built automatically from his behaviour is transformed into an ontological user model. Reusability and flexibility of the model are enhanced by introducing a novel approach to logging, which preserves the semantics of logged events. The successive analysis is driven by specialized rules, which map usage patterns to knowledge about users, stored in an ontology-based user model.

(4) Machine Learning Approach: A new robust and fuzzy relational clustering techniques using artificial immune system based on Genetic Algorithm (GA) has been proposed by [132]. The technique allows Web usage clusters to overlap, that can detect and manage the outliers in the data set via Dynamic Immune Learning approach by clustering and tracking dense evolving patterns in massive changing datasets. There has been a lot of research work done in terms of the *fuzzy logic* (FL) approach. The researchers [122] have introduced fuzzy logic in their study of the role of user profiles in Web retrieval processes, including creation, modification, storage, clustering and interpretation. In their experiments extended profiles containing additional information related to the user were presented, that personalized and customized the retrieval process as well as the website.

In another work [54], the researchers have applied fuzzy association rules to the Web log files for constructing user profiles and achieving personalization. Apart from

these, the use of a neuro-fuzzy model, to discover and analyse useful knowledge from Web log data has been proposed by [181]. In this approach to mine traffic patterns based on past trends, a Self Organizing Map (SOM) is generated, and a fuzzy inference system is used to capture the chaotic trend to provide short-term and long-term Web traffic trend predictions.

Another similar work [65] uses a machine learning algorithm for extracting and building automated user profiles. In this research, profiles are built from Web logs, and within each user profile, interesting documents are ranked according to user interests on the topic. The role of Concept Hierarchies, Compression, and Robust User Profiles for personalization has been discussed in [133]. The researchers learned that post-processed and robust user profiles have better quality than raw profiles.

(5) Cognitive Theory Based Approach: In addition, [60], [61] have presented various techniques to extract user profiles, and have introduced comprehensive user profiles that include user perceptual preference characteristics based on cognitive study. Besides, the traditional user profile extraction approach, they combined other user characteristics including cognitive approach, which is related to the learning process, experience, emotional process and visual attention.

2.2.2.1 Applicability of Clustering Methods in User Profiling

Clustering has been an important part of Web personalization, especially in collaborative filtering methods where similar users or objects are grouped to find inter-similarity of users or objects. Clustering helps in identifying similar users and objects, where their related information is then used to build various individual, group and object profiles.

Personalization approach that uses clustering when building user profiles is very popular. In one such work that adopts a CF approach, similar users are found out by clustering the access path of Web users [189]. It is Path Clustering based on Competitive Agglomeration (PCCA). The path similarity and the centre of a cluster are defined for the proposed algorithm. The algorithm relies on competitive agglomeration to get best cluster numbers automatically.

Clustering used for Personalization was done in [109], where the researchers modelled click through data as a tripartite graph consisting of users, queries and concepts embodied in the clicked pages. A Dynamic Agglomerative-Divisive

Clustering (DADC) algorithm for clustering the tripartite click through graph to identify groups of similar users, queries and concepts is developed. Due to the dynamic nature of click through data, the authors proposed the use of DADC clustering of the graph incrementally, in contrast to other models, which re-cluster the whole data. Large clusters may not be able to represent instance similarities more effectively.

Thus in order to avoid generating large clusters, DADC avoids generating large clusters using two interleaving phases, the agglomerative and divisive phases. The agglomerative phase iteratively merges similar clusters, whereas, the divisive phase iteratively splits large clusters into smaller clusters to maintain the coherence of the clusters and restructures the existing clusters to allow DADC to incrementally update the affected clusters.

Most of the CF-based personalized systems use clustering to find similar users [128], therefore the intrinsic details of such clustering methods are discussed. Most clustering methods use a distance measure to evaluate the similarity between data instances. K-means clustering is one of the best-known and commonly used algorithms. K-means [120] were inherently developed to deal with numerical data, where distances between instances are a factor for clustering them together. The widely used distance measure functions adopted by K-means are Euclidean, Manhattan and cosine. Several K-means extensions have been developed to cluster categorical data [77], [156]. Authors in [156] developed an efficient algorithm which clusters categorical data using the K-means concept. A dissimilarity function based on simple matching which evaluates the dissimilarity between a categorical instance and the cluster representative was used. The frequencies of all attributes of the instance matching the cluster are used for calculating the dissimilarity. Another approach based on K-means to cluster categorical datasets [77] used a simple matching scheme, which replaced means of clusters by modes and uses frequency to solve the best clustering outputs.

A further classification of similarity evaluation for categorical data based on neighbourhood [9], [67], [154] and learning algorithms [58], [62] is discussed in [23]. Mostly neighbourhood based evaluation methods use similarity methods as adopted by the overlap measures [23]. Some of them are Eskin, Goodall, IOF, OF, Lin, Burnaby [23]. Unlike the overlap measure, these measures consider both similarity and dissimilarity between instances, assigning 1 for a perfect match and arbitrary

small values for a mismatch.

Rock [67] and Cactus [58] are some of the popular agglomerative hierarchical algorithms which are used for categorical data clustering. Rock clusters instances in an agglomerative way maximizing the number of links in a cluster whereas Cactus utilises co-occurrence of pairs of attributes values to summarise the data and to achieve linear scaling.

Another agglomerative hierarchical algorithm [39], uses maximal entropy values between instances to cluster, however as also pointed out by [113] the complexity of it is proportional to the square of the number of instances. Birch [190] and Coolcat [18] are other popular clustering methods used for clustering categorical data. Birch uses a balanced tree structure (CF tree), which preserves the attribute relationships within different instances as leaf nodes, and then clustering is done on these leaf nodes. Coolcat is an incremental algorithm, which achieves clustering by trying to minimize the entropy values between clusters.

One approach [149] to cluster categorical and mix data uses a distance-based similarity metric similar to the one as proposed by [80]. The authors adopt a weighting scheme, which utilizes the relative importance of each instance. Once distance (d_p) between two instances X_i and X_j is evaluated, a modified version of similarity metrics defined by [88] as $S(X_{i,}X_j) = 1 - d_p(X_i, X_j)$ is used to find instances similarities.

Simple similarity measures such as overlap suffer from the problem of clustering dissimilar instances together when the number of attributes matched is the same, but attributes that are matched are different. When using sophisticated measures, finding a two-way similarity and dissimilarity is the extra overhead that has to be kept in consideration. Moreover, these similarity measures may perform well with categorical data, but in the case of mixed data, which contains both numerical and categorical data the performance declines as the complexity within clusters increases. They do not have any definition to handle and incorporate such numeric, attribute information.

2.2.2.2 Applicability of Association Rule Mining in User Profiling

Association rule mining is the process of finding those rules that satisfy the preset minimum support and confidence from a dataset. It is a two-step process, where

usually, the first step is to find frequent item sets whose frequency exceeds a specified threshold and the second step is to generate rules from those frequent item sets using the constraint of minimum support and confidence. Sequential rules are a specific category of association rules where orders of associations are also considered. The first association rule mining algorithm called AIS was proposed by [7].

The AIS algorithm has several drawbacks, notable among these are generation of too many infrequent item sets and the poor efficiency of rule generation. To improve the AIS algorithm, the researchers [7], proposed *Apriori* algorithm and later many *Apriori-based* algorithms and variations were proposed. The *Apriori* algorithm is one of the most commonly used algorithms to generate association rules.

When searching for information online, a user often searches the same content [3] many times within a website. Identification of such users, who show similarity in their search behaviour, is crucial. Such users can be recommended items based on the searches as done by the other members of the group. In case of an individual user, identification of frequent patterns of his searches can be helpful in building individual profiles.

Thus, in the case of an individual user, finding frequent patterns from user's searches can lead to the identification of a user's interest. Similarly, in the case of a group of users, identification of frequent item sets can lead to the most popular item sets, as searched by the group members. Since frequently searched item sets can be effectively mined using association rules, therefore association rule mining is an important part of many existing Web-based personalized systems.

In their work [182], the researchers focussed on identifying individual patterns in sequences of Web search log data of users. Their idea was to use such patterns for personalizing the website. A new algorithm to identify weighted maximal frequent sequential patterns from all such sequential rules was proposed. In the first step, frequency of frequent single items was used to find the weights of frequent sequences. Next, the frequent weighted sequence was defined, to find important maximal sequences.

Many recommendation frameworks based on non-sequential models such as association rules, and sequential rule models were proposed. As an example of making effective recommendations to Web users, the researchers [89] described two types of indirect associations rules, partial indirect associations and complete indirect associations. The former respect single transitive pages, while the latter covers all

existing transitive pages. An algorithm named as IDARM was proposed. The algorithm extracts complete indirect association rules with respective confidence based on pre-calculated direct rules. Both direct and indirect rules are combined as a set of complex association rules, which are used in recommendation of objects.

2.2.2.3 Two-Dimensional Vector and Matrix-based Methods

When building individual user profiles, the searches made by each user have to be compared, to find the user-items relationships. Similarly, when creating group profiles, the searches made by users have to be compared with each other to find the users-users relationships. The users-users comparison is crucial to determine which users have similar interests. In both these cases, most of the existing methods use two-dimensional methods to find these relationships between users and objects [128].

Two-dimensional data analysis techniques mostly arrange data in a vector or matrices format for finding knowledge. Vectors are rows and column arrangement of data. A matrix is a vector representation method where the rows and columns of a matrix subsequently can denote the two modes of a vector. Matrices are denoted by bold capital letters (e.g. **A**, **B**) and the rows and columns of a matrix have been denoted as $(I \times J)$ with ij signifying the i^{th} row number and j^{th} signifying the column number.

Most of the existing Web personalization systems use two-dimensional data analysis techiques to build user profiles and find relevant information from the interest vectors of users or use such methods to find users-users and users-items similarities [128], [172]. The Users data is arranged in matrix format and then matrix decomposition is employed to mine knowledge from this information. Some popular and widely used matrix decomposition techniques are (a) Singular Value Decomposition (SVD), Semi Discrete Decomposition (SDD), Independent Component Analysis (ICA), Principal Component Analysis (PCA) and Non-Negative Matrix factorization (NNMF).

In the absence of any benchmark for measuring performance of personalized systems and as this thesis focusses on building personalized systems with MDD methods like tensors, therefore the proposed methods have been compared with most popular existing two-dimensional methodologies such as SVD, PCA and NNMF.

These methods are mostly used for finding users-items and users-users correlations which are then used for building user profiles and personalized systems.

(a) **SVD (Singular Value Decomposition)**: It transforms a data matrix in order to find the hidden relationships or latent features. The transformation occurs in such a way, that the variance in data and inter dependency between relative features is exposed to a maximum. Given any matrix $\mathbf{A}$ with n rows and m columns, the SVD is given as $\mathbf{A} = USV'$ or $\mathbf{A} = U\Sigma V'$. If $\mathbf{A}$ is of rank r, then U is $n \times r$, S or Σ is a diagonal matrix of size $r \times r$, which has non-negative, decreasing elements along the diagonal (also called as singular values) such that these diagonal entries have a relationship expressed as $\sigma_1 \geq \sigma_2 \geq ... \sigma_r$ and V' is $r \times m$, Here the matrices U and V are orthogonal, such that $U'U = I$ and $V'V = I$. If matrix $\mathbf{A}$ has rank r then r is the number of non-zero entries in the diagonal matrix $(\Sigma)s$, or in other words, rs is the number of singular values of $\mathbf{A}$. The whole SVD process is explained in Figure 2.3 below.

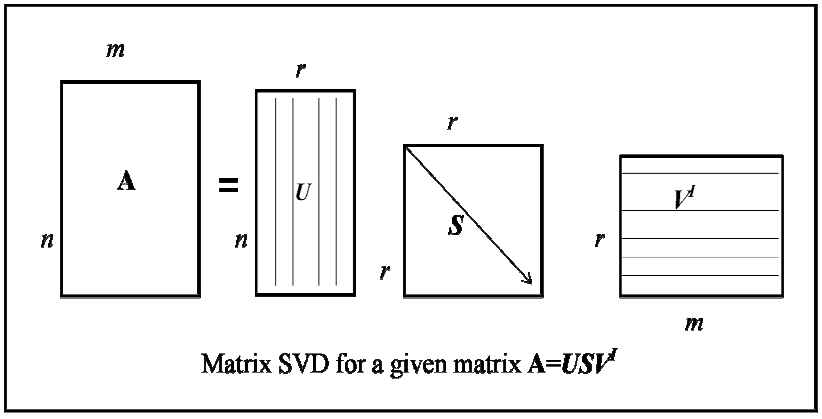

Figure 2.3 Visualization of a matrix SVD.

(b) PCA (Principal Component Analysis): PCA is an orthogonal linear transformation, which transforms the data to a new coordinate system. The transformation is such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component). The second greatest variance is on the second coordinate (called the second principal component), and so on. PCA finds the best linear projections of a set of high-dimensional points that can optimally transform given data in terms of minimum least squares costs. For

the most part principal component analysis computes the eigenvectors of the $\mathbf{AA}'$ matrix that describes the correlations between objects or the matrix $\mathbf{A}'\mathbf{A}$ which describes correlations among attributes. The first principal component accounts for the maximum variability and the subsequent variance decreases with the other subsequent components.

PCA has many similarities with SVD but there are a few limitations with PCA. PCA computes both the correlations between objects and correlations between attributes independently. On the other hand SVD computes them together represented by the two left and right singular matrices. The other problem with PCA is that computing $\mathbf{AA}'$ is difficult to handle as it is $n \times n$ and makes it more expensive to evaluate [165].

(c) NNMF (Non Negative Matrix Factorization): The inherent concept behind NNMF was to deal with datasets with no negative attributes. The belief that the negative entries in a matrix do not have any significance was the key motive behind NNMF decomposition technique. Given a matrix $\mathbf{A}$ with the size of $m \times n,$ NNMF decomposes it into $A = W \times H$ (Figure 2.4) where $\mathbf{W}$ is a matrix of factors of size $m \times r,$ and $\mathbf{H}$ is the mixing matrix of the size $r \times n,$ and $r \leq m.$ Both $\mathbf{W}$ and $\mathbf{H}$ contain non-negative entries.

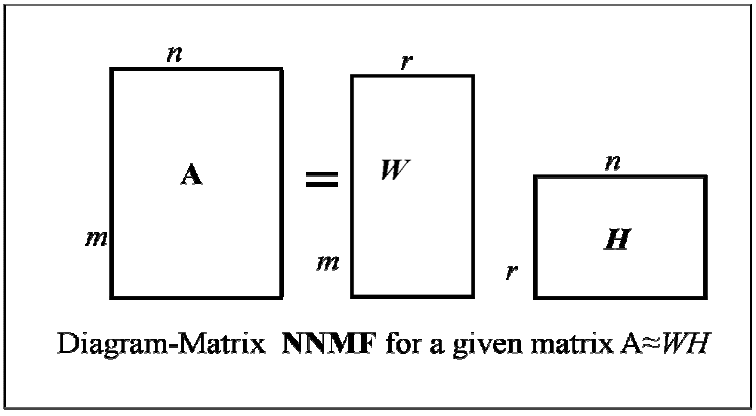

Figure 2.4 Visualization of a matrix NNMF.

NNMF has been widely used in word document analysis, face detection techniques and micro-array analysis. Nevertheless, there are a few drawbacks with NNMF. Providing a maximum number of iterations is one drawback as the convergence behaviour of NNMF is not well understood [165]. The other is deciding on the number of suitable r components. Unlike other matrix decompositions, it does not

construct one component at a time, so it is difficult to decide what a suitable choice of components would be [165].

Some researchers studied the applicability of two-dimensional methods used for building a user profile included a collaborative filtering method to provide an enhanced recommendation quality derived from user-created tags [93]. Collaborative methods of tagging items are employed to find users preferences for items. Data cubes consisting of three 2-dimensional matrices (User–item, User–tag and Tag–item frequency), which are transformed from three-dimensional space for collaborative tagging, are used. For recommendation, the Naïve Bayes classifier is used. The performance of such an approach was ascertained as being far superior to the plain collaborative recommendation approaches.

A collective model-based approach which builds customer profile model is proposed by [135]. User profiles consisting of individual user information and group behaviour information are built. A customer model reflecting product features, individual behaviour information, and the behaviour information of other users is used to build these profiles. Recommendations are done based on the customer model. Experimental results show that the proposed model has a better recommendation performance than existing models such as collaborative filtering and CF with and without product classification data. However, one major limitation is that the latent relationships between products features are ignored as two-dimensional methods are used. The other drawback is that the data for constructing group user profiles is considered demographically based on age, sex and education rather than on interest. As a result, the benefits of collaborative filtering methods may not be able to be utilized.

Another method that adopts two-dimensional techniques to build various profiles is [10], where a hybrid recommendation methodology for an online retail store is proposed. The method adopts six steps for recommendation: product taxonomy formation; grain specification; product category attributes extraction; user (customer) profile creation; user–user and user–product similarity calculation and recommendation generation.

2.2.2.4 Multi-Dimensional Methods

Multi-way data analysis refers to the extension of two-way data analysis techniques, where analysis of data of higher than two orders and n^{th} order is made. Multi-way analysis is used to extract hidden structures or patterns and capture underlying correlations between various variables in a multi-way array [102]. In the sixties Tucker [177], used such MDD modelling in chemometrics to trace the composition and effect of various components present in chemical compounds. One of the most popular techniques of representing two-way datasets is matrices. Two most popular and widely used tools for analysis of such data represented as matrices are PCA (Principal Component Analysis) and SVD (Singular Value Decomposition). These analysis tools may be very useful and powerful, but in certain scenarios, it becomes inefficient to use them.

In many research areas (such as social, neuroscience, process analysis), it has been proved that the underlying information content of data may not be captured accurately or identified uniquely by two-way analysis methods [1]. Such methods e.g. the factor models, suffer from rotational freedom unless specific constraints such as statistical independence, orthogonality, etc. are enforced [1]. In contrast, these constraints requiring prior knowledge or unrealistic assumptions are often not necessary for multi-way models. For example, in fluorescence spectroscopy data analysis [13] a Parallel Factor Analysis (PARAFAC) model can uniquely identify the pure spectra of chemicals from measurements of mixtures of chemicals. It is able to identify chemical compositions in detail due to the restrictions imposed by the model. The most significant restriction is that factors in different modes can only interact factor wise. The interaction between factors in different modes are represented by core array structures in multi-way models [1]. This core array and the subsequent component matrices are able to summarize most precisely the latent correlationships between the different factors.

Thus, it can be said that multi-way analysis has advantages over two-way analysis in terms of uniqueness, robustness to noise, and ease of interpretation in certain cases when the data has more than two dimensions. Since matrices have only two dimensions, there arises a need to represent certain data multi linearly so that the underlying relationship between each dimension independently or in relation to others can be analysed in a most efficient manner. This representation and subsequent

analysis should highlight the hidden relationship between dimensional attributes and keep information loss at a minimum.

Tensor preliminary: This thesis has followed the conventional notation that is adopted by many previous researchers [1], [55], [96], [171]. Scalars are denoted by lowercase letters, e.g., c. All vectors are represented by boldface lowercase letters e.g., $\boldsymbol{v}$. The i^{th} entry of $\boldsymbol{v}$ is denoted by $\boldsymbol{v}_i$. Matrices are denoted by boldface capital letters, e.g., $\mathbf{A}$. The j^{th} column of $\mathbf{A}$ is denoted by a_j and element (i, j) by a_{ij}. Tensors are denoted by boldface Euler script letters, e.g., $\boldsymbol{\mathcal{X}}$. Element (i, j, k) of a 3rd-order tensor $\boldsymbol{\mathcal{X}}$ is denoted by x_{ijk}.

A vector is a one-dimensional data array and can be referenced or accessed using a single index. E.g. a_i. Matrix is a two-dimensional data array consisting of some arbitrary values for each row and column entries. These values in a matrix can be referenced by a two-digit index e.g. $A_{i,j}$, i for the row, and j for the column entry position of each element in **A**. Similarly a tensor is a multi-dimensional data array which has $1 \ldots n$ dimensions. The order of a tensor is the number of dimensions, also known as ways or modes. E.g. the tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{M_1 \times M_2 \times \ldots M_n}$ has dimensions from $1..n$ Vectors and matrices can be thought of as tensors of order one and two respectively. All vectors are tensors, but not all tensors are vectors. The first two dimension of a tensor can be thought of as similar to a matrix of dimension $\mathbf{A}_{m1,m2}$. This representation property of Tensor can be useful when unfolding a tensor into a matrix. E.g., for the given tensor $\boldsymbol{\mathcal{T}}$ as above, it can have matrices of size $[\mathbf{M}_1 \times \mathbf{M}_2]$ for the first two dimensions and the product of the remaining dimensions $[\mathbf{M}_3 \times \mathbf{M}_4 \times \mathbf{M}_{n-1} \times \mathbf{M}_n]$ would define how many matrices of size $[\mathbf{M}_1 \times \mathbf{M}_2]$ are to be created to represent the full structure of a tensor in each dimension as in a matrix form.

Each element of a tensor needs n indices to represent or reference its precise position in a tensor. E.g. the element a needs a_{ijklmn} indices where *I, J, K, L, M, N* are the dimension of a tensor. Visualization of tensors up to the 3rd dimension is easy but it becomes confusing to understand the structure of tensors of a higher order than three. For instance, Figure 2.5(a) below shows a representation containing three dimensions of a tensor denoted by X, Y, Z-axis. Each axis represents exactly one dimension. For

example, axis X represents all car makes (e.g. Ford, Toyota), Y represents all car models, and Z represents all car body types (e.g. sedan, hatch).

Figure 2.5(b) below shows three searches made by one user. The cube alongside would be a tensor space where all such search values, compromising these three dimensions would lie. For multiple objects or users, a tensor could be imagined as slices of a cube as in Figure 2.5(c) where each slice represents a distinct user's complete detailed search information and each slice is similar to Figure 2.5(b). The Figures 2.5 and 2.6 further help to visualize the structure of a third and fourth order tensor.

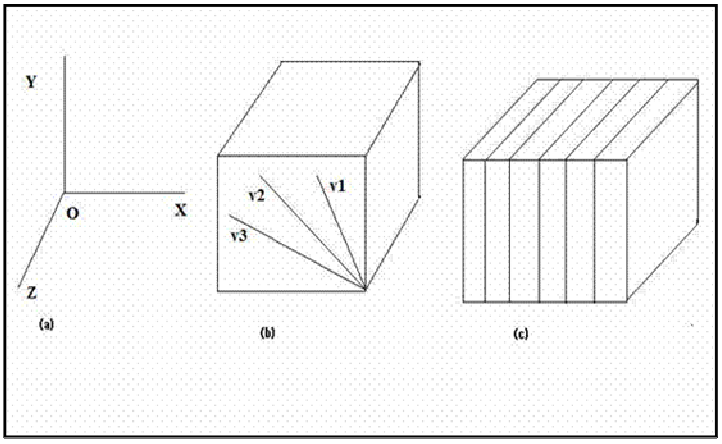

Figure 2.5 A Visualization of 3rd order tensor.

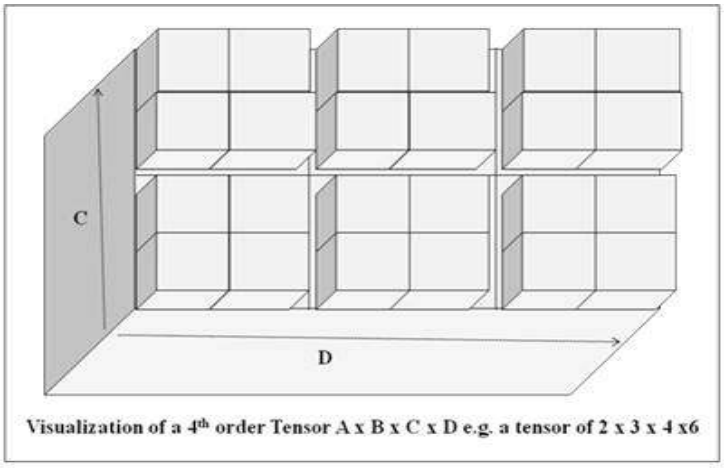

Figure 2.6 A Visualization of 4th order tensor.

Properties of tensors:

Definition 1. (Addition): Given any two tensors $\mathcal{D} \in \mathbb{R}^{I \times J \times K}$ and $\mathcal{H} \in \mathbb{R}^{I \times J \times K}$ the sum (*S*) is a tensor $\mathcal{S} \in \mathbb{R}^{I \times J \times K}$ evaluated as $\mathcal{S} = \mathcal{D} + \mathcal{H}$, and element wise as $s_{ijk} = d_{ijk} + h_{ijk}$.

Definition 2. (Multiplication): Multiplication of a tensor $\mathcal{B}$ with a scalar α is a tensor $\mathcal{P}$ and is expressed as $\mathcal{P} = \alpha\mathcal{B}$ or element wise as $p = \alpha b_{ijk}$.

Definition 3. (Outer Product): The outer product between two tensors $\mathcal{P} \in \mathbb{R}^{I \times J \times K}$ and $\mathcal{B} \in \mathbb{R}^{L \times M \times N}$ is an order six tensor $\mathcal{C}$ with $\mathcal{C} \in \mathbb{R}^{I \times J \times K \times L \times M \times N}$ ∋ $\mathcal{C} = \mathcal{P} \circ \mathcal{B}$, or element wise as $c_{ijklmn} = p_{ijk} b_{lmn}$. The symbol o denotes the outer product of vectors. This is also the generalization of the outer product between any two vectors **p** and **b** which result in pb^{T}. The vector outer product is defined as follows. Let a, b and c be column vectors of size $I \times 1$, $J \times 1$ and $K \times 1$ and $\mathcal{P}$ is a tensor of size $I \times J \times K$, then $\mathcal{P} = \mathrm{a} \circ \mathrm{b} \circ \mathrm{c}$ if and only if $p_{ijk} = a_i b_j c_k$.

Definition 4. (Inner Product): The inner product (also called the dot product or scalar product) between two tensors $\mathcal{D}$ and $\mathcal{H}$ having same dimensions is a scalar as is denoted as in Equation (1).

$$\langle \mathcal{D}, \mathcal{H} \rangle = \sum_{ijk} d_{ijk} h_{ijk} \tag{1}$$

Tensors whose scalar product is 0 are mutually orthogonal.

Definition 5. (Kronecker product): The Kronecker product denoted by symbol ⊗ for two matrices **A** and **B** is **(A⊗B).** It multiplies each a_{ij} by matrix **B** or in other words, it creates many copies of matrix **B** and scales each one by the corresponding entry of **A**. The Kronecker product of matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{s \times t}$ represented by matrix $\mathbf{K_1}$ here is denoted as matrix $\mathbf{A} \otimes \mathbf{B} = \mathbf{K}_1 \in \mathbb{R}^{ms \times nt}$.

Where

$$\mathbf{A} \otimes \mathbf{B} = \mathbf{K_1} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2n}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & a_{m2}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix} \neq \mathbf{B} \otimes \mathbf{A} = \mathbf{K_2} = \begin{bmatrix} b_{11}\mathbf{A} & b_{12}\mathbf{A} & \cdots & b_{1t}\mathbf{A} \\ b_{21}\mathbf{A} & b_{22}\mathbf{A} & \cdots & b_{2t}\mathbf{A} \\ \vdots & \vdots & \ddots & \vdots \\ b_{s1}\mathbf{A} & b_{s2}\mathbf{A} & \cdots & b_{st}\mathbf{A} \end{bmatrix}$$

Definition 6. (Hadamard product): The Hadamard product of two matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{m \times n}$ of same number of rows and columns is denoted by $\mathbf{A} * \mathbf{B} \in \mathbb{R}^{n \times m}$. It is evaluated as shown in Equation (2).

$$\mathbf{A} * \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \cdots & a_{1n}b_{1n} \\ a_{21}b_{21} & a_{22}b_{22} & \cdots & a_{2n}b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{m1} & a_{m2}b_{m2} & \cdots & a_{mn}b_{mn} \end{bmatrix} \tag{2}$$

Definition 7. (Khatri Rao Product): The Khatri-Rao Product (denoted by $\odot$) of two matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{l \times n} s$ is denoted by $\mathbf{A} \odot \mathbf{B} \in \mathbb{R}^{ml \times n}$. To calculate the Khatri-Rao product, the two matrices should have the same number of columns. The Khatri-Rao product can be viewed as a column-wise Kronecker product. The Khatri-Rao product of two vectors is identical to the Kronecker product of two vectors. For the two matrices **A** and **B** the Khatri-Rao product is represented as

$$\mathbf{A} \odot \mathbf{B} = [\mathrm{a}_1 \otimes \mathrm{b}_1 \quad \mathrm{a}_2 \otimes \mathrm{b}_2 \ldots\ldots \mathrm{a}_n \otimes \mathrm{b}_n]$$

Definition 8. (Dimension/Mode Product): The mode n-product of a tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ by a matrix $\mathbf{M} \in \mathbb{R}^{J_n \times I_n}$ is denoted by $\mathcal{T} \times_n \mathbf{M}$ is an $(I_1 \times I_2 \times \ldots \times I_{n-1} J_n \times I_{n+1} \ldots \times I_N)$ tensor defined as in Equation (3).

$$(\mathcal{T} \times_n \mathbf{M})_{i_1 i_2 \ldots j_n \ldots i_N} = \sum_{i_n} t_{i_1 i_2 \ldots i_n \ldots i_N} m_{j_n i_n} \tag{3}$$

for all index values.

Definition 9. (Rank 1 tensors): An N^{th} order tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ has rank 1 when it equals the outer product of N vectors $U^{(1)}, U^{(2)}, \ldots, U^{(N)}$:

$$\mathcal{T} = U^{(1)} \circ U^{(2)} \circ \ldots \circ U^{(N)}.$$

Rank of a tensor: The rank of an arbitrary tensor $\mathcal{T}$ of N^{th} order is denoted $\mathrm{R} = \mathrm{rank}(\mathcal{T})$, is the minimal number of rank one tensors that yield $\mathcal{T}$ in a linear combination.

Norm of Tensor: Norm is the scalar measure of the magnitude of a vector or matrix and is denoted by $\|\mathbf{A}\|$. Norm for a vector could be the Euclidian norm and is evaluated as $\|x\|_{\mathrm{E}} \equiv (\sum_{\mathrm{i}=1}^{\mathrm{n}} |x_{\mathrm{i}}|^2)^{1/2}$. Matrix norm is defined as $\|\mathbf{A}\| \equiv \max(\|\mathbf{A}\mathbf{x}\| / \|\mathbf{x}\|)$ where ***x*** is any non null vector. The Frobenius norm of a matrix is defined as in Equation (4).

$$\|\mathbf{A}\|_F \equiv [\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2]^{1/2}. \quad (4)$$

The norm of a tensor is given as the square root of the sum of the squares of all its elements, i.e., for a tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ of size $I \times J \times K$, it is defined as $\|\mathrm{X}\|^2 \equiv \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} x_{ijk}^2$. This is the higher-order analogue of the matrix Frobenius norm. The Frobenius norm of a tensor $\mathcal{T}$ is also defined as $\|\mathrm{X}\| = \langle \mathrm{X}, \mathrm{X} \rangle^{1/2}$.

Diagonal Tensors: A tensor $\mathcal{X} \in \mathbb{R}^{M_1\times M_2\times \ldots M_N}$ is diagonal (Figure 2.7) if $x_{m_1} x_{m_2} \ldots_{m_N} \neq 0$ and only if $m_1 = m_2 = \ldots = m_N$.

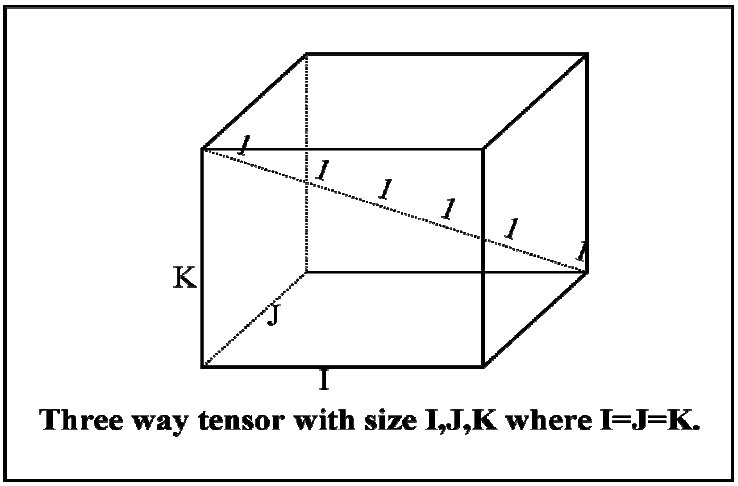

Figure 2.7 A three-dimensional diagonal tensor with ones along the super diagonal.

Symmetry of Tensors: A tensor $\mathcal{X} \in \mathbb{R}^{M_1\times M_2\times \ldots M_N}$ is cubical if every dimension/mode is the same size, i.e., $M_1=M_n$. A cubical tensor is called super symmetric if its elements remain constant under any permutation of the indices [96]. A three-way tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$, given with *I=J=K (and let I, J, K=N)* is super symmetric only if all the elements are the same in all dimensions under all permutations of indices. Thus it is super symmetric only if $x_{ijk} = x_{ikj} = x_{jik} = x_{jki} = x_{kij} = x_{kji}$ for all $i, j, j = 1, \ldots, N_{(I=J=K)}$.

Matricization of a tensor: It is also known as *canonical tensor matricization*. In various tensor decomposition techniques the dimensions are flattened to represent matrices of various sizes before the subsequent decomposition technique is applied. Matricizing, unfolding or flattening of a tensor is a useful operation for transforming a given multi-dimensional array into a matrix. A third order tensor $\mathcal{T} \in \mathbb{R}^{I\times J\times K}$ is able to

form three matrices of $I \times JK, J \times IK, K \times IJ$ and each matrix has columns as shown in Figures 2.8 below.

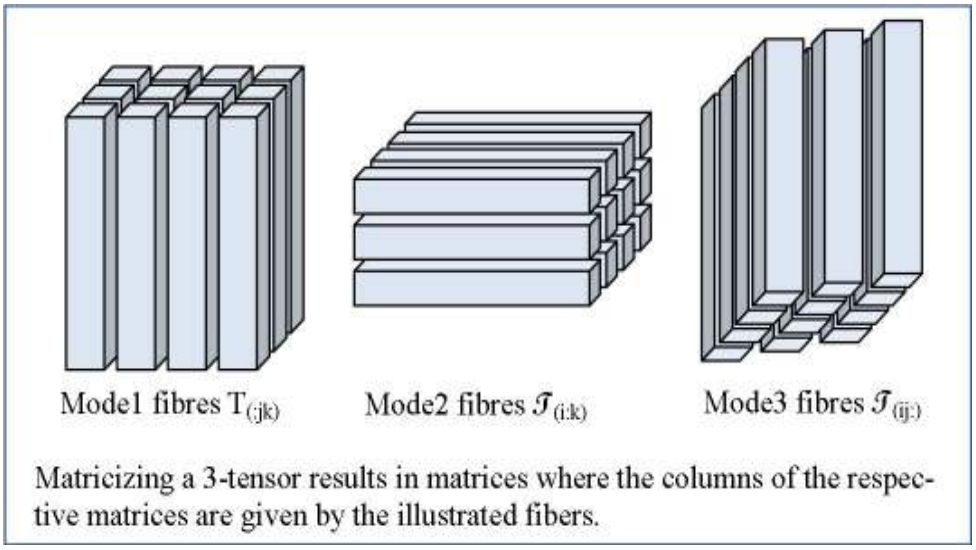

Figure 2.8 Matrix unfolding of a tensor with vertical, horizontal and lateral slices.

Matricizing is an important operation of tensor flattening [95]. The following Figures 2.9 and 2.10 further help in the visualization of how matricizing occurs. Given a third order tensor $\mathcal{T} \in \mathbb{R}^{4\times4\times6}$ the matricizing can also be done based on grouping individual component matrices.

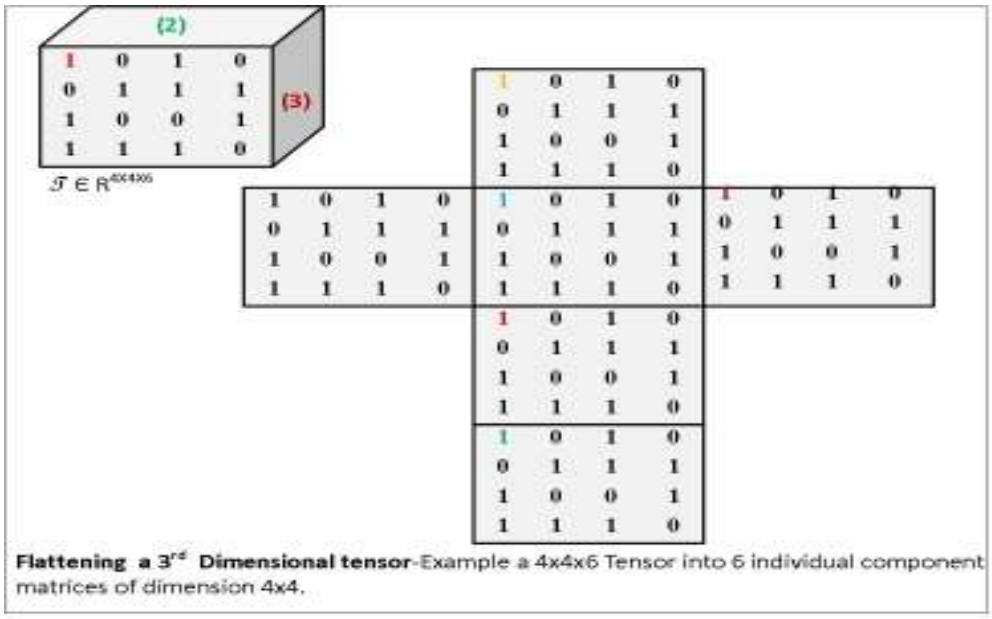

Figure 2.9 View of tensor with various component matrices.

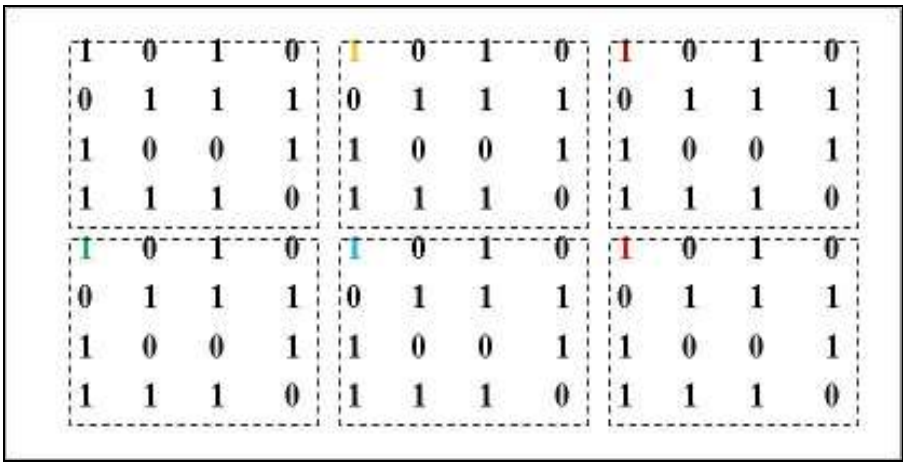

Figure 2.10 An example of a tensor flattened as a single matrix.

Popular Tensor Decomposition Techniques: Decomposition helps in data reduction, keeping the relationship between dimensions intact. It captures the multilinear structure in data, by analysing the factors in each dimension. These factors are linear combinations of the variables in that dimension. Multilinearity of the model means that the structure of the model is the same in each dimension or in other words, it means that the model is linear in each dimension. The overall influence and correlations of factors in each dimension is then represented by a component matrix, whose columns are the factors determined by the model. The constructed matrix summarizes the structure in each dimension.

The two most well known and commonly used multi-way models are Tucker models [177] and the PARAFAC model [68], which is also called CANDECOMP (Canonical Decomposition). CANDECOMP [33] was proposed independently but is considered equivalent to PARAFAC. A new tensor decomposition model based on matrix SVD was also proposed by [47]. It is known as HOSVD (Higher Order Singular Value Decomposition). Besides these three most popular tensor decomposition methods, various moderations of tensor decomposition techniques, which have originated from the PARAFAC and Tucker family, also exist. These techniques are HOOI, INDSCAL, PARAFAC2, CANDELINC, DEDICOM, and PARATUCK2. The three important decomposition methods are briefly discussed below.

a) Tucker Decomposition Technique: Tucker models are also known as N-way principal component analysis techniques [177]. The main objective of Tucker decomposition (Figure 2.11) is to approximate a large tensor using a small tensor

through a change of basis. A Tucker decomposition form as defined by [96] on tensor $\mathcal{X}$ is shown in Equation (5).

$$x \approx [\![\mathcal{G}; A^{(1)} .... A^{(N)}]\!]. \tag{5}$$

The tensor $\mathcal{G}$ is a core tensor, which is a generalization of the full tensor to the desired rank. If $\mathcal{G}$ is of size $J_1 ... J_n$, then each matrix $\mathbf{A}^{(n)}$ formed after is of size $I_n \times J_n$.

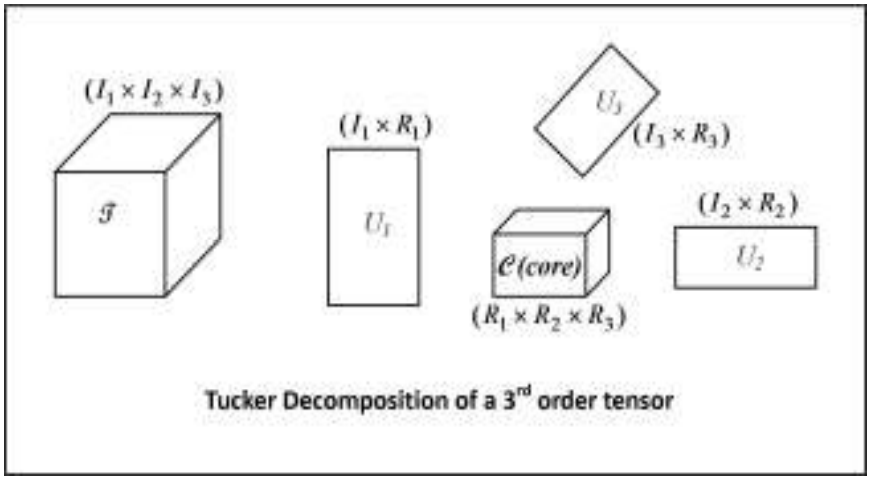

Figure 2.11 Tucker decomposition of a 3rd order Tensor results in component matrices as shown.

Definition 10 (Tucker Decomposition). Given an input tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \ldots \times n_m}$ and core tensor sizes $\{r_1, ... r_M\}$, find a core tensor $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_m}$ and a sequence of projection matrices $\mathrm{U}^{(\mathrm{d})} \big|_{d=1}^{M} \in \mathbb{R}^{n_d \times r_s}$ such that $\left\| \mathcal{X} - \mathcal{Y} \times_1 \mathrm{U}^{(1)} \times \ldots \times_M \mathrm{U}^{(M)} \right\|$ is small, i.e., $\mathcal{X} \approx \mathcal{Y} \times_1 \mathrm{U}^{(1)} \times \ldots \times_M \mathrm{U}^{(M)}$.

If $\mathrm{U}_{(d)}$ is full rank for $1 \le d \le M$, and $r_d = n_d$ the approximation can be exact. Tucker decomposition changes the bases for every dimension/mode through the projection matrix $\mathrm{U}^{(d)} \big|_{d=1}^{M}$.

Various adaptations of the tucker model like Tucker1, Tucker2, Tucker3 have come up. Tucker1 is based on the idea of rearranging data as a matrix and decomposing the matricized data using SVD. Both Tucker2 and Tucker3 models allow rank reduction in more than one dimension and are named after the number of modes in which rank reduction is allowed. When compared with a PARAFAC model a Tucker3 model is a more flexible model. This flexibility is due to the core array ($\mathcal{C}$), which allows an

interaction between factors in different modes. The core array explores the underlying structure of a multi-way dataset much better than a restricted PARAFAC model. However as pointed out by [1], the full-core array structure of Tucker3 model has a few drawbacks. Firstly, the interaction between factors in different dimension causes rotational indeterminacy in Tucker3 models. Unlike PARAFAC, a Tucker3 model does not determine component matrices uniquely. When a component matrix is rotated by a rotation matrix, it is possible to apply the inverse of the rotation matrix to the core and still obtain the same model. Therefore, a Tucker3 model can determine component matrices only up to a rotation. Secondly, interpretation of Tucker3 models is much more difficult compared to PARAFAC models [1].

The following are the properties of Tucker decomposition.

1. Dimension/Mode n-multiplication: $x \approx y \times U^{(1)} \times .... \times_M U^{(M)}$.
2. Matricization: $\mathrm{X}_{(d)} \approx \mathrm{U}^{(d)} \mathrm{Y}_{(d)} (\mathrm{U}^{(M)} \otimes .... \otimes \mathrm{U}^{(d+1)} \otimes \mathrm{U}^{(d-1)} \otimes ... \otimes \mathrm{U}^{(1)})^{\mathrm{T}}$.
3. Outer Product: $\mathrm{X} \approx \sum_{i_1=1}^{n_1} \sum_{i_2=2}^{n_2} ... \sum_{i_M=1}^{n_M} \mathrm{y}_{i_1, i_2, ... i_M} \mathrm{u}_{i_1}^{(1)} \circ \mathrm{u}_{i_2}^{(2)} \circ ... \mathrm{u}_{i_M}^{(M)}$.

b) PARAFAC Decomposition Technique: PARAFAC [68], also known as Parallel Factors or better known as CANDECOMP is a generalization of PCA (Principal Component Analysis) to higher order arrays. It is based on Cattell's principle of Parallel Proportional Profiles [34]. PARAFAC aims to obtain a unique solution such that component matrices are determined uniquely up to a permutation and scaling of columns. This uniqueness property is what which makes PARAFAC a popular technique in many fields [1]. Mathematically, a tensor decomposed by PARAFAC model (Figure 2.12) can be represented as a linear combination of rank-1 tensors (Where an N^{th} order rank-1 tensor is a tensor that can be written as the outer product of *N* vectors) [1]. However all the component matrices of a PARAFAC model, in general, cannot satisfy orthogonality constraints. To decompose a tensor with a PARAFAC model, which will give component matrices with orthogonal columns, a tensor should be diagonalizable. However, with tensors in general this is not the case [95]. Various methods are used to determine the number of factors in a PARAFAC model, which include residual analysis, visual appearance of loadings, number of iterations of the algorithm, core consistency [26], etc. The core consistency factor diagnostic model resembles a combination of a Tucker3 core and a super-diagonal PARAFAC core. However, there is no sure-fire method of determining the optimal

number of factors (in terms of interpretation) for real data. Therefore, it is often suggested that several diagnostic tools should be used together instead of using one single method. Given a tensor of rank 3 as in $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ then a R-component PARAFAC decomposed model can be represented as

$$\mathcal{X} = \sum_{r=1}^{R} a_r \circ b_r \circ c_r + E,$$

or equivalently element wise as in Equation (6).

$$x_{ijk} = \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr} + E. \tag{6}$$

where a_{ir}, b_{ir}, c_{ir} are the i^{th} column of component matrices $\mathbf{A} \in \mathbb{R}^{I \times R}, \mathbf{B} \in \mathbb{R}^{J \times R}$ and $\mathbf{C} \in \mathbb{R}^{K \times R}$ respectively and $E \in \mathbb{R}^{I \times J \times K}$ is the array containing residuals. x_{ijk} represents an entry of a three-way array of $\mathcal{X}$ in the i^{th} row, j^{th} column and k^{th} tube.

Property 1 (PARAFAC). The following presentations are equivalent,

1. Mode-n multiplication: $x \approx \mathrm{y} \times \mathrm{U}^{(1)} \times \ldots \times_M \mathrm{U}^{(M)}$ where y is a super-diagonal tensor with the diagonal entries $\lambda_1 \ldots \lambda_r$.
2. Matricization: $\mathrm{X}_{(d)} \approx \mathrm{U}^{(d)} \Lambda (\mathrm{U}^{(M)} \odot \ldots . \odot \mathrm{U}^{(d+1)} \odot \mathrm{U}^{(d-1)} \odot \ldots \odot \mathrm{U}^{(1)})^{\mathrm{T}}$. where Λ is a diagonal matrix with diagonal entries $\lambda_1 \ldots \lambda_r$.
3. Outer product: $\mathrm{X} \approx \sum_{i=1}^{r} \lambda_i \mathrm{u}_i^{(1)} \circ \ldots \mathrm{u}_i^{(M)}$.

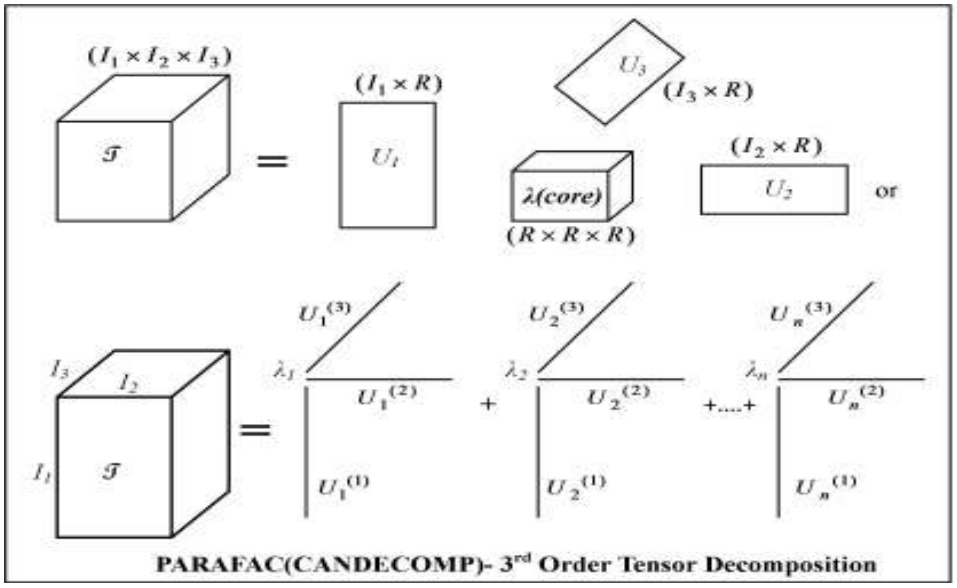

Figure 2.12 PARAFAC decomposition of 3rd order tensor gives component matrices as shown.

c) Higher Order Singular Value Decomposition: The methodology to decompose higher order matrices into low rank approximation using singular value decomposition on such component matrices was proposed by [47]. Tucker3 with orthogonality constraints on the component matrices has been named as Higher-Order Singular Value Decomposition (HOSVD). HOSVD can be computed by flattening the tensor in each dimension and calculating the singular vectors (SVD) corresponding to that dimension, which are also called *n-mode* singular vectors (Figure 2.13). However, unlike SVD, HOSVD does not provide the best rank-$(R_1, R_2, ..R_N)$ approximation of a tensor [47], where R_i is the rank of tensor in i^{th} dimension. The rank of a tensor in n^{th} dimension/mode is called $n-rank,$ which is the dimension of the vector space spanned by the columns of the matrix obtained by flattening the tensor in n^{th} dimension.

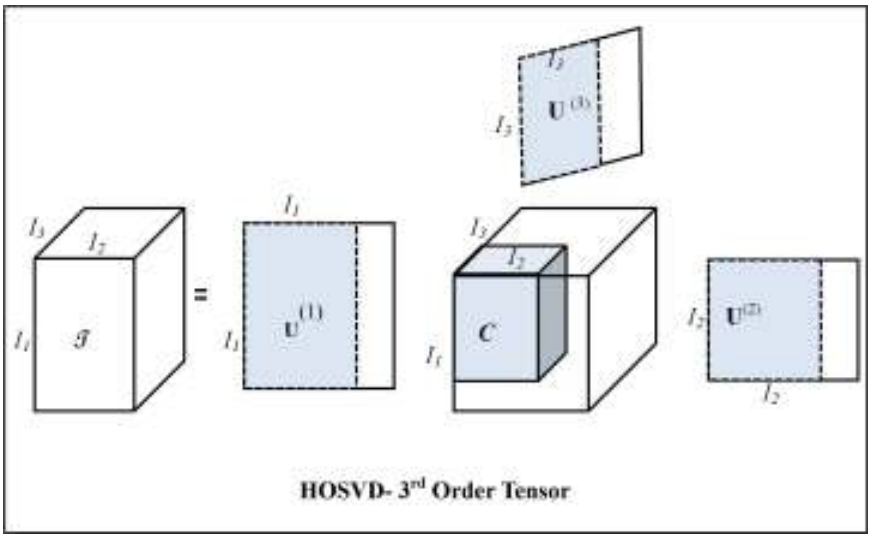

Figure 2.13 HOSVD decomposition of 3rd order tensor gives component matrices as shown.

Various Other Tensor Decomposition Techniques: There are a number of other tensor decomposition techniques which have been classified mostly into PARAFAC and the Tucker family based on the parental behavioural and similarity to these two methods. Some other well-known techniques derived from the three prominent classes of tensor decomposition families the PARAFAC, Tucker and HOSVD are briefly described in Table 2.2.

Name	Derived Family	Description	Year Proposed/ Proposed By
INDSCAL	PARAFAC	Individual Differences in Scaling	1970/[33]
PARAFAC2	PARAFAC	Parallel Factors for Cross Products	1972/[69]
DEDICOM		Decomposition into Directional Components	1978/[70]
CANDELINC	PARAFAC	CANDECOMP with Linear Constraints	1980/[51]
TUCKALS3	Tucker	Tucker Alternating Least Square	1980/[103]
PARATUCK2	PARAFAC, Tucker	PARAFAC and Tucker2	1996/[71]
PARALIND	PARAFAC	Parallel profiles with Linear Dependences	2005/[25]
Constrained-block PARAFAC	PARAFAC	Constrained-block PARAFAC	2006/[46]
HOOI	HOSVD	Higher Order Orthogonal Iteration of Tensors	2007/[162]

Table 2.2 Some other tensor dimension reduction techniques.

Applicability of Tensors in Various Scientific Fields and Web Mining: Tensors mathematics is one of the features in multi-linear algebra that has the capability to address issues related to multi-way data modelling and analysis methods. Tensor modelling has been used extensively in chemometrics and psychometrics and has been gaining a lot of momentum and popularity in other domains such as computer vision, neurosciences, signal processing, time series, anomaly detection, numerical analysis, data mining, graph analysis and many more.

Tensors have been used in the identification of handwritten digits [159], face identification [142], [175] image/objects patterns identification applying multilinear principal component analysis (MPCA) on tensor object models [144]. The other prominent work includes the work of [52], who have used tensors for finding similarity between aspects of a documents such as similarity of abstract, of title ,of keywords, of authors and of citations. They use data from eleven SIAM Journals and SIAM proceedings for years 1999-2004, and decompose the resulting 5022 x 5022 x 5 dataset matrix using CP tensor decomposition. A few examples of usage of tensors of three dimensions represented as first order, second and third order is shown in Table 2.3.

Application	First order	Second order	Third Order
Identification of Handwritten digits	Pixels	Samples	Classes
Electronic noise data	Sensors	Time	Gases
Network Sensor data	Source-IP	Destination-IP	Port
Computer vision	Persons	Pixel	View/illumination
Bio-informatics	Genes	Expression Variable	Micro array
Chat Room Analysis	Users	Keywords	Time Windows
Author Similarity	Authors	Timestamp /Year	Publications /Keywords
Patient/Disease Patterns	Patients	Symptoms	Diseases
Web Link Analysis	Words	Web pages	Links

Table 2.3 Some examples of applications of a 3rd Order tensor.

Applicability of Tensors for Data Mining the WWW: The application of tensors in Web-based personalization applications is still in its infancy, barring a few prominent works such as [172]. The researchers analysed the click through data of users by using HOSVD decomposition to analyse search from click stream data in order to personalize Web search. In this work, the researchers represented click through data as a 3rd order tensor and proposed a tensor decomposition approach based on the generalization of the matrix Singular Value Decomposition (SVD). The application of the Tucker3 decomposition was applied to chartroom data, by [2]. Here the researchers used TSM to analyse user behaviour in chat rooms. To achieve this objective three dimensional data from chartroom activities such as users, keywords and time windows was taken. From their analysis, the researchers found that tensor decomposition was appropriate for such MDD data due to the number of components in each dimension. Additionally, using tensor decomposition rather than two-dimensional decomposition they found that interaction patterns in the data was advantageous to mine.

Another interesting work is that of [97], where the researchers tried to find the relationship of Web pages with the words, documents and links. These hyperlinks between pages were based on authorities, and hub, and links were ranked based on these two derived variables. Their results discovered authorities; sites with higher scores in a component. Recently some recommendation models, which use three-dimensional tensors for recommending music, tags and objects, have been proposed. Recommender models, using HOSVD for dimension reduction, have been proposed

for recommending personalized music [153] and Tags [173]. Researchers [148] have used a TSM-based tag recommendation model which uses tensor factors by multiplying the three features matrices with core matrix each consisting of user, items and tags

2.2.3 Summary: User Profile

Capturing appropriate user behaviour based on his previous browsing habits is a difficult task to achieve. User behaviour changes over time and the multi-dimensional nature of searched data further worsens the situation, as traditional two-dimensional methods are used to mine knowledge from search datasets. To identify the browsing patterns of a Web user's behaviour a wide range of techniques ranging from AI, data mining, psychology and information theory have been studied and used. However as mentioned by [127], [128], [130], for effective user personalization, everything depends solely on creation of good quality and useful '*aggregate usage profile'* of users from the patterns mined.

Most of the existing methods use two-dimensional models to build user profiles, where a user's interests are represented as a vector and subsequently similarity between users is measured by comparing these vectors using some distance measure or using matrix-based methods such as SVD, PCA, NNMF etc. However, due to the MDD nature of Web data such methods may not be able to generate effective user profiles. Representing the MDD search log in two-dimensional space limits the freedom of various dimensions to interact freely and effectively with each other and thus much of valuable information is lost.

One of the other major challenges in user profiling is the implementation of adaptive methods in profile building so that user profiles can automatically derive relevant information from the available data sources [134]. Since different users may have different objectives while browsing pages within a website, a detailed study of all the possible objectives and a well-structured lay-out of the domain knowledge to be able to infer the mapping of a user's objective has to be determined. An example is if a user is a seller who wants to sell his car, he may also have searched for new cars in the same website. Thus, in this case his profile may have two contrasting values. Thus, the user may have two profiles like the buyer and seller. Therefore,

identification of a profile category is important in creating appropriate profiles of a user.

This research focuses on building a particular category of profiles only however, such profile building methods could be extended to other user profile categories. Furthermore, the similarity methods adopted by such profile-building methods, need to consider integrating various sources of information and the appropriate mapping of subsequent dimensions [128] of each source for extracting the best information to build user profiles. This research utilizes MDD methods to build user profiles.

2.3 Recommendation Systems

Human judgement is often biased by others and sometimes, it often becomes difficult for some people to choose exactly what they want. In a scenario such as purchasing a car people will often prefer to have a recommendation or another's opinion before making a decision. For example, when choosing clothes, music or movies or when buying electronic goods or cars, people often look for suggestions. Such suggestions are in fact recommendations, and the impact of these recommendations depends on the mutual relationship or common interests the two people share. For the ease and assistance of users in their search, most websites recommend items, articles or consumer goods, which are the most popular or have been searched by many Web users.

Thus, Web recommendation is the electronic way of giving referrals to users when they are searching for products or services online. The *Tapestry* system [66], *GroupLens* [150], *Ringo* [161] and *Lotus Notes* [121] were some of the early systems making recommendations to users. The *Tapestry* and *GroupLens* were used to recommend news and articles to users based on their interests, whereas *Ringo* was used to recommend music to users based on world of mouth approach (i.e. ratings). *Lotus Notes*, based on the concepts of pointers, adopted a CF approach, by which interesting hyperlinks with some contextual information about the source were shared among various *Lotus Notes* users.

2.3.1 Recommendation Methods in Context of World Wide Web

One of the most successful and one of the earliest pioneers of using recommendation in their website is *Amazon.com*. It recommended books and other popular searched items to its visitors based on collaborative filtering techniques. *Amazon* provides dozens of forms of personalization features. Some popular ones are *Your Amazon*, *Today's Deals, Gifts & Wish Lists, Recommendations by Category, Your Browsing History, Your Lists,* and *Your Profile*. Following the footsteps of *Amazon* are other websites such as *Eachmovie, Movielens, MovieFinder* and *Netflix* that used collaborative filtering methods to recommend movies. As well, there are websites such as *Audioscrobbler, CDNow, iLike, iTunes, Last.fm, MusicMatch, MSN Music, MyStrands, RealPlayer MusicStore, Rhapsody*, and *Napster* to recommend music, while *TiVo* and *tvgenius.net* recommend television shows, and *Findory* recommends news. Another very popular e-commerce website that uses recommender systems is *eBay*. It deals with various products and uses recommendations to attract its registered users. Broadly, recommendation systems can be classified into three categories as content-based, collaborative filtering (CF) and hybrid systems

1. Content-based recommendation systems, which implicitly study user behaviour and create user profiles based on the contents of documents he/she views. A user profile is nothing but ranked documents in decreasing order of preference that a user likes. Recommendations of items are based on his/her user profile. These types of recommendation methods are mostly called as content-based recommendation systems. Thus, the items recommended would often be similar to what the user has indicated they preferred or liked from past interactions with the system [126], [135]. Some other examples of such systems are the *NewsDude* [22], *News Weeder* [106] and *Infofinder* [104], which recommend to users news that they may be interested.

2. Collaborative Filtering based recommendation systems or social filtering methods have been used by websites such as (*eBay, Amazon, GroupLens, MovieLens, CDNow.com, Levis.com, Moviefinder.com*, and *Half.com*). Such CF recommendation models use the collective information of user, who are grouped mostly based on the ratings given to items they have purchased or reviewed. A new user is recommended items in which he may be interested based on the social group of users into which he falls. This is deduced mostly by the initial reviews a user makes about some items. There can be many variants of CF-based recommendation such as demographic

recommendation**,** utility-based recommendation [29] and knowledge-based recommender systems.

Following [21], [24], [48], [49] recommendation methods build on CF techniques can be grouped in two general classes as neighbourhood and model-based methods. Neighbourhood or (memory-based [24], or heuristic-based [4] methods use Item-to-Item or User -to-Item Correlation to find the neighbours and subsequently uses this information for recommendation to its users.

Item-to-Item Correlation: Item to item adopts a content-based approach since knowledge about the products (i.e. contents) is used for recommendation and only similar matching products/contents are recommended. In these recommendation systems similar items of interest to a user or purchased by are recommended. Popular websites that use this kind of recommendation include *Reel's Movie Matches; Moviefinder's Match Maker;* and *Amazon's More such as this*. Recommender systems based on such an approach are manual, because they need user feedback before deciding what to recommend to a user.

Users -to-Item Correlation: Such systems are based on the CF approach [73], [99], [161] adopting the nearest-neighbour techniques, where the interests of a group of people are combined to find the highest rated interests and then and then interests such as items, products or people are recommended to the group. Popular and well known are *Amazon's 'What Other Customers Are Looking At Right Now'*; CDNOW's – *'Customers Who Bought This Item Also Bought'*; *Moviefinder's 'We Predict'* and '*Style Finder';* as well as *RSVP's 'People who looked at this profile also looked at'*. All of which are examples of recommending users based on the CF approach.

Neighbourhood-based methods are intuitive and relatively simple to implement. One of the strong points of neighbourhood-based systems is their efficiency and cost, as such systems do not require any costly training and modelling phases. In cases where large datasets are used, such methods can compute neighbours as an offline process thus providing near instantaneous recommendations. Fewer memory requirements make these methods more scalable to applications having millions of users and items.

Recommender systems based on this approach have the advantage that they are not much affected by the constant addition of users, items and ratings. Once an item's similarities have been computed, an item-based system can readily make recommendations to new users, without having to re-train or model the system for

each set of new users. In situations where few ratings have been entered for a new item, the similarities between this item and the ones already in the system need to be only computed to make recommendations.

Conversely, model-based recommender systems use these ratings of users and items to build predictive models, using various machine learning, data mining and other techniques and then make recommendations based on the models. The general idea is to model the users-items interactions with factors representing latent characteristics of the users and items in the system, such as the preference class of users and the category class of items [49].

Recent investigations show that, the state of the art model-based approaches are superior to neighbourhood-based ones in the task of predicting more ratings [49], [100], [174]. However, it is also pointed by [49], [72] that such good prediction accuracy alone does not guarantee users an effective and satisfying experience. On the other hand model-based approaches are superior at characterizing the preferences of a user with latent factors [49]. Neighbourhood approaches, on the other hand are able to efficiently capture local associations in the data [49]. Some popular recommenders using the CF approach are:

Pointers [121] was among the first systems that facilitated use of collaborative filtering techniques for recommendations. *Pointers* are implemented inside *Lotus Notes* environment. Users of both systems can publish and distribute the bookmarks and add the comments to the Web page. These features enabled users to actively share information with others. In the *Pointers* system a pointer consisted of URL link, contextual information, and comments by the senders.

PHOAKS : People Helping One Another Know Stuff [176] is a system that recommends the URLs that will be very interesting to users. The system automatically recognizes Web resources in a news group message, classifies it, and recommends it to other users. It scans and checks the group's messages, uses collaborative filtering to get the most important URLs in these messages. After sorting these links, the system recommends these URLs to users.

Siteseer: *Siteseer* [152] is a collaborative system that uses Web browser bookmarks to find neighbors and recommend sites. Users with significant overlap in bookmark listings are deemed to be similar, and are recommended un-visited sites within the similar group category they belong.

3. Hybrid recommendation systems combine both the techniques used by content, based recommendation systems and collaborative-based recommender systems. These recommender systems use knowledge about users and products (objects) and use collaborative filtering techniques to recommend. It identifies user needs and recommends the best option that matches his searches. The need of users can be provided explicitly by a user in opinion, surveys, reviews, ratings or implicitly collected from Web server logs (dynamic data) or user's registration data (static data).

The hybrid approach is popular, and is capable of providing with more accurate recommendations [4]. A popular example is the *FindMe* systems like *Entree* and recommender.com [28], [29], [30]. The first *FindMe* system was the Car Navigator, an information access system used for finding new car models. The system, rated cars for features like horse-power, price, gas mileage, and these ratings could directly be manipulated by users. Selections were made by users and retrieval was performed by turning the individual selection criteria into a similarity-finding query to find a new set of cars.

Another recommendation system, which is one of the earliest and popular, is the restaurant recommender '*Entrée'* [30], which makes its recommendations by finding restaurants in a new city similar to restaurants the user knows and likes. People rated the cuisine, price, quality, and atmosphere of restaurants in a ranked order, based on what was important to them. Since different people may have different preferences for each characterstics of restaurant, *FindMe* systems therefore had different retrieval strategies, each capturing a different notion of similarity.

Similarly, *PickAFlick*, a movie recommender system which created its multiple lists of similar movies by employing three retrieval strategies, one that concentrated on genre; another focused on actors; and a third that emphasized direction. These systems were popular but, user intervention was always needed in most of these systems.

Another methodology that uses a hybrid between existing memory and model based algorithms using CF techniques is [136]. Experiments are performed on *MovieLens* data (rating movies) where ratings given to movies by a user are used to determine the personality type (i.e. interests) of a user. The similarity of personality is compared with other users, and then based on this diagnosis, the probability that a user will like some new items is calculated.

Another hybrid approach which is called the meta-level hybrid approach combines two models. The output of one model is given as the input to other model and the

entire model becomes the input to another model. The first meta-level hybrid was the Web filtering system *Fab* [17]. It, worked on the concept of group profiles where documents were first collected on the basis of their interest to the community (i.e. group of people) and then were distributed to particular users. *Fab* was also performing a cascade of collaborative collection and content-based recommendation, even though the collaborative step only created pool of documents and its ranking information was not used by the selection component [29]. The other shortcoming of *Fab* was its reliance on explicit user feedback.

In addition to the hybrid recommender systems mentioned above, some other popular ones are:

GAB: Group asynchronous browsing [184], used multi-tree data structure for storing the details of bookmarks and hot listed (i.e. frequently visited or used) files of users. Sibling relationships between similar categories of topic/links were defined to improve relationships between objects stored in the tree. The whole notion of doing this was to identify and recommend similar Web pages or files to a user. In order to avoid privacy issues, as the whole system had access to a user's private data, the system provided the users with an option allow saving of their bookmarked information in the private or public space.

Let's Browse & Letizia: Let's browse [115] and its predecessor, *Letizia* [114], are Web agents that assist a user during his browsing experience. Both systems build user profiles by monitoring a user's behaviour, his browsing time and other details. The *Let's Browse* system uses a single user profile for recommendation whereas *Letizia* improves on Let's Browse by using group profiles for making recommendations.

WebWatcher: The *WebWatcher* system [86] is an interactive recommender system where a user enters his requirements into the system. However, unlike a keyword-based search engine, it employs user's and group profiles to find similarities and to make recommendations. The system also involves the user's experience to reinforce learning. The major drawback of this system is that it needs a user's input information prior to making recommendations.

One recent work that proposes a hybrid approach that uses neural nets for recommendation is [36]. The proposed approach trains the artificial neural networks to group users into different clusters, and applies the well-established *Kano's* method for extracting the implicit needs of users in different clusters.

Apart from these three popular approaches, CF, Content and Hybrid several different methodologies are used for making recommendations to individual users as well as groups of users. Some of these use an approach known as an attribute-based recommender systems as proposed by [160]. It recommends products to customers based on syntactic properties of the product.

For example, a search for an historical romance book, would recommend books based on those particular attributes. *Reel's Movie Map* is a good example of an attribute-based recommendation. Such recommendations are made entirely on the basis of the category of movie selected by users. *Movie Map* works manually by a user selecting a category each time he visits the website before any recommendation is made. However, *Amazon* remembers user's interests as it has registered users.

In Another work by [94], describes a concept-based recommender system that recommends papers to general users of the *CiteSeerx1* digital library of Computer Science research publications. User profiles based on the *ACM* (Association for Computer Machinery) classification tree are created for which past click histories are used. Based on this profile and search, relevant papers in the domain are recommended to users.

Methods for group recommendation based on relevancy is proposed in the work of [11]. The novel notion of consensus function is proposed in this work. It consists of two components, relevance and disagreement, which for each candidate item, produces a single recommendation score that is a weighted summation of the two component scores.

In this work [10], a hybrid recommendation for an on-line retail store is proposed. The methodology extracts user preferences in each product category separately and provides more personalized recommendations by employing product taxonomy, attributes of product categories, and Web usage mining.

A feature-based machine learning approach for making personalized recommendations, that is capable of handling the cold-start problem has been proposed by [42]. The method employs a model-based approach where user profiles containing details of user's interests are used along with the content profiles for making recommendations.

Limitations of various recommendation methods: With content-based recommendation systems, sometimes the limited quantity of content may not be enough to make a good analysis of the problem [4]. These systems may also suffer

from the problem of over- specialization, which is when a user is recommended items based on his own ratings. Since, the system will recommend items scoring highly against the user profile, unique or different products cannot be recommended [108]. A user's profile information or rating information limits his access to other interesting information as liked by other similar users.

A knowledge-based recommender system does not have a *'ramp-up'* problem [28], where items rated highly by users are given preferences since, its recommendations do not depend on a base of user ratings. However, the main drawback of knowledge-based systems is its dependence on knowledge acquisition methods which are a well-known bottleneck for many artificial intelligence applications [4]. The other problem with content-based recommender systems is that in most of the cases these systems need some user intervention for making recommendations.

The CF-based-recommender systems are size dependant; meaning the performance of system is directly dependant on the data [172]. The more data, the better the probability of finding an exact or very close match for a new user. However, there is the problem of scalability with such methods. Recommendation systems usually handle very large data profiles to form the neighbourhood, hence the nearest neighbour algorithm is often very time-consuming and scales poorly in practice [40], [135].

Another drawback of recommender systems based on the CF approach is that a '*ramp-up'* situation is created when a sufficient number of users rating data are not available to identify the user's likings and disliking. In these situation recommendations based on the CF approach will not be very useful for an individual user [28]. Other major problem with CF-based recommendation methods is the cold-start problem. Collaborative filtering systems provide little or no value when a user is the first one in his neighbourhood to enter a rating for an item [157].

Furthermore, in order for any machine learning methods used in recommendation [28], to learn good classifiers needs a sufficient number of individual ratings of items by users. The New User problem (i.e. recommending to a new user) can be somewhat resolved by CF methods but the same problem with different perspectives exists when a new item is introduced or needs to be recommended.

2.3.1.1 Applicability of Clustering Methods in Recommendation

Clustering similar users has been a popular approach and has used extensively by many websites in their recommender systems such as *Amazon, eBay, Netflix, Eachmovie, MovieLens, MovieFinder, Audioscrobbler, CDNow*, *iLike, iTunes, Last.fm, MusicMatch, MSN Music, MyStrands, RealPlayer MusicStore, Rhapsody*, and *Napster*. The most common and widely used methodology adopted by these websites and their recommender systems is the identification of a group of users who have similar interests. Recommendations are made based on the similarity of items being searched; or the similarity of search behaviour; or both.

A personalized CF approach using clustering for personalized search improvement using the similarity of query selection; desktop information; and explicit relevance judgments across people grouped in different ways, is proposed by [176]. The groupings explored fall into two dimensions, the longevity of the group members relationship, and how explicitly the group is formed. Results by such personalized search methods clearly showed that similar people grouped into the same groups have better personalized results delivered to them. Nevertheless, finding the best groups implicitly is a difficult task.

In another study of recommendation, a hybrid Web personalization system based on clustering and contiguous sequential patterns mining is proposed [182]. The system clusters log files to determine the basic architecture of websites, and for each cluster, a contiguous sequential pattern mining to further optimize the topologies of websites is instigated.

2.3.1.2 Applicability of Association Rule Mining in Recommendation

When making recommendations for either an individual user or a group of users, association rule mining has been a popular approach. In case of an individual user, searches made by him can be analysed to find interesting associations of objects. This information can then be utilized to give him recommendations. In the case of group of users, associations between their searches are ascertained and such associations can then be used to make recommendations between users of the same group. Some recommendation methodologies that have adopted this approach are described below.

In one of the methodologies [118], a novel hybrid recommendation method that combines the segmentation-based sequential rule method with the segmentation-based KNN-CF method is proposed. A sequential rule-based recommendation method analyses customers purchase behaviour over time to extract sequential rules. The method uses customers RFM (Recency, Frequency, and Monetary) values to cluster similar users. A customer's previous buying behaviour is also analysed, apart from mining the sequential rules. These are rules that are extracted for each group from the purchase sequences of that group to make recommendations. Consequently, the segmentation-based KNN-CF method provides recommendations based on the target customer's purchase data for the current period. The results of the two methods are combined to make final recommendations.

A genetic algorithm that formulates purposeful association rules out of the transactions database of a transportation management system has been proposed by [107]. The constructed rules are recommended to the associated users. The recommendation process takes into account the constructed rules and techniques that are derived from collaborative filtering.

2.3.2 Summary: Recommender Systems

Recommender systems are a useful alternatives to search algorithms since these systems help users discover items (movies, music, books, news, Web pages, friends (in social networks)), that they might not have found by themselves otherwise. Some of the challenges when designing recommender systems are:

1. **Handling limited**, **missing and obsolete data:** One of the most prevalent problems of CF-based recommender systems is the absence of the rating data of users. Users often fail to provide rating of items that they may like, as it seems an extra effort and waste of time on their part. To overcome this problem this research focuses on implicit data acquisition methods (e.g. server logs) to track the interests of users.

2. **Recommending items with many features or items with multi-dimensional properties:** When a user has made many searches that are unrelated or are different, in these cases, it becomes difficult to mine knowledge from such datasets. When traditional two-dimensional methods are used to find similarity between users and

items, latent relationships that exist between users, searched items and their features may be lost. Therefore, to overcome these drawbacks, this thesis proposes the use of tensors to mine a user's interest in various dimensions, and then recommendations are based on the *top* Υ, interests of a user in that dimension.

3. **How to overcome new/anonymous user, new item/object problems effectively, as such problems are commonly faced by many recommender systems:** To recommend objects to a new user, this thesis proposes to use information in the group profile and object profile. The searches made by a user are compared with the *top* Υ searches in each group, and group objects that match a user's search closely are then recommended. In cases where no ratings are available for an object (new item problem) unlike previously used methods objects can either be modelled using MDD technique or clustered based on their properties. When a new object is searched by a user, other objects, which are in the cluster and match closely to the objects as searched by a user, can be recommended.

4. **How to make unique recommendations that may interest a user:** To make unique recommendations that may interest a user, apart from his individual profile, the group profile can be used to make recommendations. To recommend interesting objects which are similar to a user's preferences, a user can be recommended objects that are rated highly in the group profile he belongs to.

5. **In the case of personalized recommender systems, how to use various profiles and object profiles in synchronization:** The three profiles created user, group and object can be used independently or in conjunction to make recommendations. Each profile building method adopts a different strategy. Individual user profiling is a model-based approach, which identifies key interests of a user in each searched dimension, and then recommends objects based on his profile. The group profiling method employs a hybrid approach, which utilises knowledge about the objects (represented as searches) and the users (CF approach).

Another approach utilizing similar methodology as adopted by attribute-based recommender systems [42], [160], is also proposed in this thesis. Object profiles consisting of similar objects are created based on object properties. A user searching for a particular object/item in a category can be offered similar or closely related

objects in that category. These objects should have the highest similarity between them. Keeping this in consideration, this thesis focuses on using MDD and novel clustering methods to group similar objects.

6. **Deciding best recommendations given to a user and how such recommendation should be given to a user (whether ranked or plain *top* ϒ recommendations):** Most recommender systems do not use all available information for making recommendations. When making recommendations, a user's current preferences; his past browsing behaviour; similar objects as searched by other users (CF approach); are seldom all used for making recommendations. Thus, to make recommendation more effective, this thesis proposes that the order of selecting information should be in the following priority order.

a) User's current preferences: All recommendations should be based on keeping a user's current preferences at the highest level.

b) Past browsing behaviour: Previous searches analysed from individual profiles.

c) Similar objects as searched: Analysed from object profiles.

d) Searches made by similar users: Analysed from group profiles.

Once the best ordering of preferences is achieved based on the type of user, such recommendations can be scored, based on the preferences matched. The results can thus be displayed in order of priority with closely matched recommendations at the top.

7. **Scalability, real time performance of model based recommender systems and evaluation metrics for such systems:** CF-based systems suffer from scalability problems [40], [135]. However, scalability is not an issue when most of profile building task like clustering [179] and indexing [5] is done offline. Checking the real time performance of recommender systems is unfeasible and costly as it needs a big structure, where considerably large numbers of users are asked about the feedback on recommendations given to them by the personalization system. Therefore, to evaluate the performance of the recommender systems, this thesis proposes to use training data, based on which user profiles are constructed, and then compare the recommendations made by the various methods with the subsequent actual searches made by users.

2.4 Ranking Methodology

Most Web systems have methods to identify users. Some use registration information, while others may use cookies to track users. User profiles built by these Web systems contain information about a user's interests and can be effectively used for returning good quality searches and the most similar search results matching a user's need [6], [139]. However, personalized systems based on user profile alone cannot be helpful in returning best recommendations and search results. The recommendations need to be ranked/ordered to keep a user's current preferences at highest level.

When a user makes a search the results that are returned are often ranked based on the number of query parameters matched with the user's search. In some cases profile information is used for ranking these results, and in many cases CF methods are employed, which rank highly searched objects higher on the scale [48], [93], [128], [157]. Methods that employ all available information for ranking recommendation and search results are still uncommon.

2.4.1 Ranking Web Search and Recommended Results

Once, a user directs a query to a website, various information processing agents in the form of individual, group and object profiles work in the background to filter information, which can then be recommended to a user. Due to the sheer size and volume of databases, retrieval of relevant information as needed by users has become a cumbersome process. Information seekers are faced with the problem of information overload - too many result sets are returned for their queries. Moreover, too few results or no results are returned if a specific query is asked. Thus, even in the case when such external information processing agents are used, there is a need to rank the result sets according to the preferences of a user.

A user query indicating search requirements is usually comprised of many features (or parameters or attributes). A relevant result set is retrieved by matching all or some search parameters, regardless of how closely or not they match the level of importance of a user's interests. For each user, different searched parameters may hold different importance. If the user is known (e.g. a registered user) then based on his past behaviour, it is possible to find out what features interest him most.

However, for a new user, it is difficult to discover the intention behind the search. One option available for giving relevant information to such users is to utilize collaborative filtering. Unlike most Web-based personalized systems, that only employ user profiles to make a recommendation, user profile information has to be coupled with a ranking method that utilizes the profile information to rank highly relevant results higher.

2.4.2 Ranking Methods in Context of World Wide Web

Seldom is the ranking of recommendation results discussed in the context of Web personalized systems, as most of the existing systems use popular or common ranking methods. In this section, some of the earlier work for ranking of query result sets is discussed. In general database/Web ranking methodologies are motivated by the two communities; the IR [8], [36], [37], and the Machine Learning Community [6],[187]. From time to time, different researchers belonging to both these communities use workload [8], [37] and user profile information [6], [139] to improve the quality of results given to a user.

Ranking models motivated by the IR community are mostly workload dependant and widely use TF (Term Frequency), IDF (Inverse Document Frequency) or a combination of these to rank and score database results. Many previous researchers have pointed out the drawbacks of ranking methods based on existing vector methods, which use TF-IDF as a weighting scheme. One of the biggest disadvantages of using TF-IDF is that query results may be biased in favour of ranking highly searched results higher [111], [188]. This can result in neglecting users actual search needs. Both the techniques QF (Query Frequency) and OFIDF (Query Frequency Inverse Document Frequency) as adopted by [8] are workload dependant. From the workload, similar occurring attributes are measured using the Jaccard coefficient. The similarity function thus obtained is also inspired by the TF-IDF. The other disadvantage when using QF and IDF is that for *many-answers* problem (i.e. too many results are returned, therefore finding relevant information becomes difficult) many tuples may get the same score [8] and for *empty-answers* problem (i.e. no results or not enough relevant results are returned) the results beyond the top scored tuples may be quite similar. The importance of attributes based on workload is defined in order to solve

this problem, however, using IDF to find any similarity between tied tuples with missing attributes, it again creates problems in both cases when tuples with high IDF and low IDF are used to score [8].

The combined metrics called QFIDF [8] may work well in most of cases, however in cases such as when a query is entirely different or is unrelated to the previous workload, QFIDF ranking may fail to give good quality rankings. The other drawback with QFIDF is that in case when not many values are referenced by a user (when searched features are relatively fewer compared to the available features), a non zero score may bias the results, which will result in poor quality of rankings relative to a user's query.

In another significant work [37], the researchers applied probabilistic information retrieval model-based approach for ranking database query results. The effective solution to the problem of *many-answers* has been proposed. However, as pointed out by the authors themselves, the implementation of such a ranking function is too complex, tricky and involves many correlations between data values.

Some prominent work in this area was done in [38] where the authors have proposed a novel method of ranking top *k*-rows that match the user's requirement. Their methodology revolves around a scoring variable, which is used to retrieve top-*k* query results. If this variable is high, no rows are retrieved and if low, too many rows are retrieved. Other algorithms [101] have all adopted a nearest neighbour approach with many variations to rank the best result sets returned to a user. In case only a few features match a user's query these methods may not be able to give good ranking scores. Too much dependency on the scoring variable is another limitation of such methods.

The other community using machine learning approaches to rank database tuples uses various data mining approaches from KNN (k-nearest neighbour) [101] to classification [187] and various machine learning approaches such as neural net [6] to rank database tuples.

Another KNN approach as proposed by [187] uses a BM25 model to find top k-ranked documents or results for a given user's query. KNN approach is used to find similar documents relative to the user's query and finally a mean of features from these top k ranked documents is taken. However, one major problem with the KNN approach is that when two different users issue two similar queries, the ranking given by such a model would have similar results for the two users, regardless of the

intentions of the two users. This can happen as the training model would have same *k-nearest neighbour* for such queries. The other drawback of KNN methodology is deducing the exact or optimal number of k for each query. Furthermore, as shown in the experiments in [187], the query classification (QC) based approach performs better than the single model approach (where no distinction is made between query types). However, categorization of every query and than evaluation of most similar results from a set of such queries seems to be an unfeasible option. The other drawback is that sometimes most of the query criteria given by users are either short or ambiguous and a wrong categorization of such queries may give contrasting results as expected by the users.

In another work [6] the researchers discussed incorporating user profile information to increase efficiency of searched results. The authors used the supervised machine learning technique (neural net tuning algorithm) to learn a ranking function that best predicts the relevance of each search result.

Apart from these query ranking methods an ontology based search by utilizing user profile information has been proposed by [139]. The authors have utilized three sub techniques such as re-ranking, filtering and query expansion to collectively improve the performance of recommended documents. Re-ranking depends on a ranking function, which is applied to results returned by a search engine. Filtering eliminates mostly all irrelevant documents and it is done by comparing the documents to a list of keywords or previous ratings provided by the users. Query Expansion is achieved by modifying the query to include the user's interests in the query.

Some recent significant works related to the ranking of user's queries was completed by the researchers [166], who proposed a probabilistic ranking model based on partial orders to rank result sets in the case where lots of values are missing or are uncertain. In another work [194], a database keyword-based search is proposed by indexing the related information about tuples. However, there is the extra overhead of creating, managing and updating indexes that such a system has to handle. In the study done in [124] the authors have proposed a method to give most approximate results to a Web user based on his query. Methods of tackling *empty-answers* have been proposed, such as re-writing the user's query and relaxing constraints.

2.4.3 Summary: Ranking Methods

This section of the thesis discussed the various ranking methodologies, adopting different similarity functions to give a ranked list of documents or objects to a user. Most of the existing Web personalized systems do not utilize all available information for ranking the recommendations. In a personalized search environment, objects that are being searched by users may have a mix of categorical and numeric attributes. A ranking methodology that caters to numeric, categorical, and mixed attributes is needed. Besides matching the searched attributes as searched by a user, it should be able to give personalized search results. Thus, such a methodology should have the capability of handling mixed data types and returning result sets according to the user profile. In the case where no such information is available, the methodology should have the ability to return results based on the CF approach. Overall, a function which is simple and flexible, is easy to implement, utilises available profile information to give optimum results, and which has the ability to deal with mixed data types is needed.

2.5 Research Gap: Web Personalization Systems

After studying and analysing the various literatures this research work identified some research gaps in relation to Web-based personalized systems and related techniques. Undoubtedly, personalization for each Web user would change the way people seek information online. However as discussed by [128], [169], [172] when comparing users-items for making personalized recommendations such systems need to consider the multi-dimensionality of user-items and keep the inherent relationships intact. At present it is mostly two-dimensional vector and matrix-based methods that are used to find similarity between different users [128], [172]. Since, most of the Web search log data consisting of users and items is multi-dimensional and traditional two-dimensional methods are employed to find relationships between users and items, personalized system built from two-dimensional methods would not be able to give recommendations of good quality.

Keeping all this in consideration, this thesis tries to emphasise on building user profiles using MDD techniques, so that there is minimum information loss across the various dimensions, the latent relationships between users and objects are captured

and as a result of this the inter user and inter object similarity between similar users and objects are maximised. Good similarity measures would make sure that good clusters of users and objects are formed. This will be helpful in making the personalized system more effective.

Apart from this, some other research gaps and issues in relation to Web-based personalized systems that still need to be addressed or improved are:

1. Unsuitability of Web personalization architecture or systems due to
    - Different programming language with various limitations and applicability of each.
    - Different techniques (Data mining, IR, Machine Learning etc) without proper benchmark used, to identify optimal methodology.
    - Different standards of metadata (Human input).
    - Dynamic nature of Internet and related architecture. E.g. Use of different technologies, whose applicability is limited to a specific domain. Eg. Mobiles, PDA, blogs, Semantic Web 2.0 or other Web Applications.
2. Inappropriate survey methods or content and limitations regarding information acquisition methods, due to security and privacy issues.
3. Identification of social groups and what role each user plays in such groups. The impact of the likes and dislikes of an individual user or a group about a product, feature, or person that influences the decisions of related group members.
4. Integration of social sciences, behavioural sciences and mathematical or computer sciences to define a methodology to achieve personalization.
5. With the growth and advent of social networks such as *Facebook, Twitter, YouTube, Flickr, Last.fm, Delicious, Twitter, Myspace, Orkut,* etc, CF-based personalization has gained momentum. Social networking websites try to exploit the relationships that different users share. These websites recommend people within a group to each other, if they are already not known to each other. The *GOSSPLE* [91] project has the positive aim of fully personalizing the search process, improving the chances of a user to find relevant content.
6. Adaptive hypermedia and adaptive Web personalization techniques which include tasks and activities; cultural preferences; social interaction etc; as well as cultural adaptivity (including multilingual personalization); dialogue and simulation personalization , [91] are other essential considerations which have to be kept in mind while future personalized systems are developed.

2.6 Summary: Literature Review

Personalization is a complex process consisting of various processes such as user profiling, recommendation, and ranking process. These three processes constitute the important components, and are the core of any personalized Web system. There has to be synchronization between the three processes, to make the best recommendations that can satisfy a user's needs. In the literature review section, various methods adopted by many researchers were discussed, and the popular approaches like CF-based recommendations, user model based recommendations and content-based personalized recommender systems were discussed.

Various approaches to achieve such personalization, using methods such as clustering; association rule mining; neural networks; machine learning-based approaches and ontology-based approaches etc were discussed. As can be seen in the literature, multi-dimensional data models have rarely been used to make such personalized systems. Users-items, items-items and users-users similarity measures mostly adopted by such personalized systems are based on two-dimensional data analysis techniques.

Thus, there is a need to represent and model users, item's similarity using some high-dimensional data analysis techniques. In most of the Web personalized systems user profiles are created without giving much importance to the latent relationships that exist between different dimensions of the user's searches. When building user models, either for an individual user or for a group of users, the model needs to relate to each dimension.

The other important component of a personalized system is the recommendation model. To achieve the best results in recommendations, that is, matching a user's query with his need, is the supreme goal of a recommender system. To achieve this, various issues such as recommending new items; recommending items to a new user; recommending based on various profiles; using profiles in synchronization and deciding on *top* Υ recommendations need to be addressed uniformly by a personalized system. The other issue is how to utilize the available information in various user profiles and map this with a user's current search to give the best recommendations. To achieve this, a ranking methodology that can enhance personalized systems by prioritising a user's current preferences with greater accuracy and that can utilize profile information is needed.

Chapter 3 The Proposed Research Methodology

In the preceding chapter the background needed to propose an effective personalization system was discussed. Section 2.2 of the thesis, presented a detailed discussion about the current user profiling techniques. Section 2.3 discussed about the recommender systems and Section 2.4 surveyed the current ranking methodologies. This chapter would introduce the proposed method of personalization and its related components- the user profile, recommender and ranking methods, using an innovative approach of multi-dimensional data modelling. Firstly, this chapter gives an overview of the proposed system. Section 3.1 begins with some preliminary information needed to create user profiles and gives an overview of the user profiling methodology. The next sub-sections then discuss about creating individual, group and object tensor models from the Web log data of users. Section 3.2 details about clustering methods. Apart from discussing general clustering, the sub-sections discuss the proposed DIF (Dimensional Influence Factor) and FIBCLUS (Fibonacci based Clustering) clustering algorithms that can enhance clustering efficiency of users and objects respectively. The next Section 3.3 and its sub-sections details about the individual, group and object profile creation methods. Section 3.4 and its sub-sections details about the recommendation methods based on individual, group and object profiles. Finally in the last Section 3.5 the proposed FIT (Feature Importance Technique) based Ranking is discussed.

In the next chapter 4, a case study of '*Mileage Cars*' the prototype website is undertaken, which identifies different profile categories and how models for each can be built. The next following chapters 5, 6 and 7 would present empirical analysis of the proposed methodology and its components.

All Web-based personalized systems have an objective, to make quality recommendations. For making effective recommendations this thesis proposes that, the various models such as user, group and objects should be built using Tensors. All model building is done as an offline process. The complete detailed research methodology is divided into six phases. Figure 3.1 provides an overview of each phase and processes involved in each of these phases. In the first phase all data pre-processing is done. In the next phase individual, group and object model building approaches are discussed. Once different models are built and decomposed, in the next phase clustering is done to find similar users and objects. In the fourth phase

individual, group and object profile construction methods are discussed. The fifth phase discusses about recommendation methods for individual and group of users. Finally, in the sixth phase, once recommendations are made to users based on the available information a ranking methodology to rank/score recommendations is discussed.

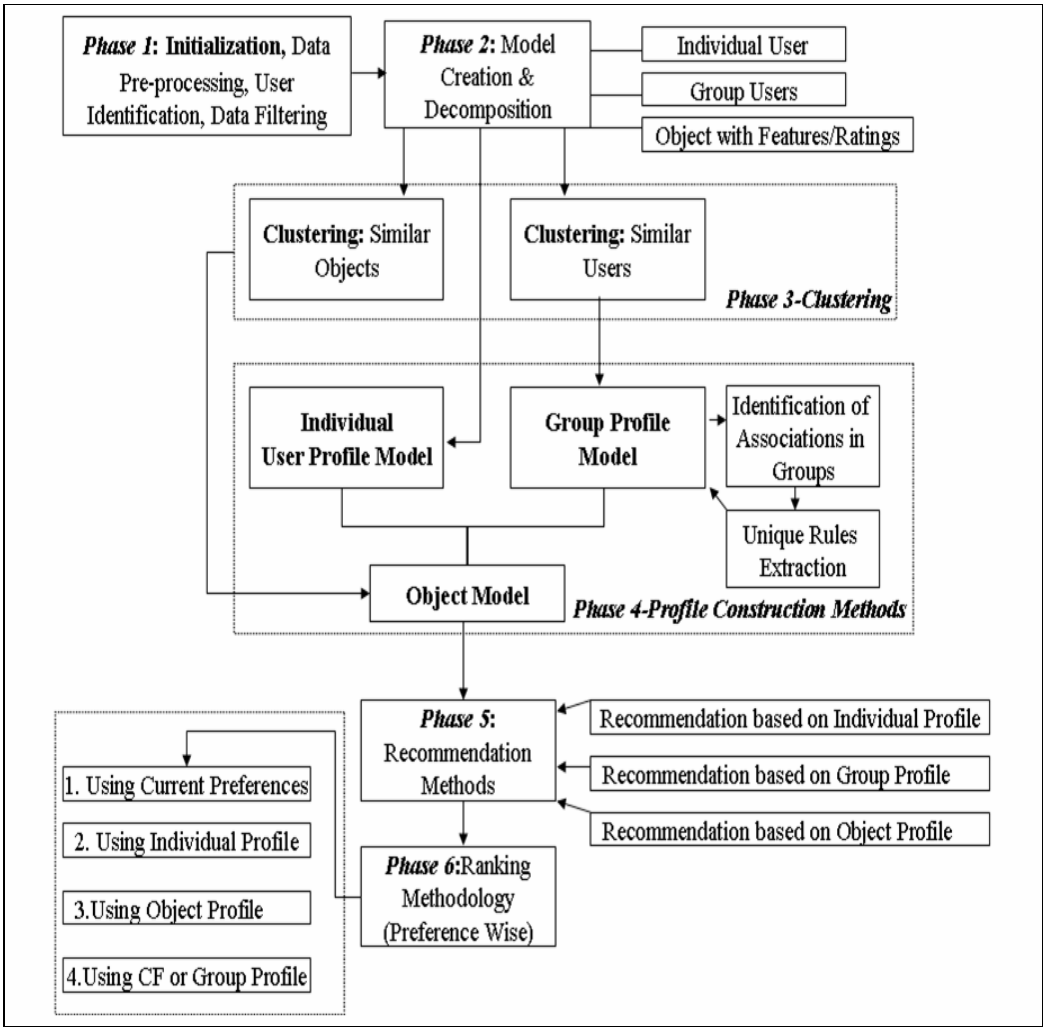

Figure 3.1 Proposed research methodology, phase-wise in detail.

The following terminology will be used throughout this work. Let $U = \{u_1, u_2, ..u_z\}$ be the set of users whose search behaviour has to be modelled. Let $\mathcal{T}$ represent the user's behaviour tensor model that map the user search behaviour represented in the form of search parameters. For a user this tensor is represented as

$\mathcal{T} \in \mathbb{R}^{M_1 \times M_2 \times .. \times M_n}$, where each dimension from $M_1 .. M_n$ represents the various mapped searched dimension values (or distinct searched features in that dimension).

An example in a car sales website such searched dimensions would be the car makes, car models, cost ranges, body types etc as searched by users. Each of the dimensions in the tensor model represents features in the parameterised search as provided by the website. Let $Iu_{j\eta}$ be the user's parameterized search queries representing his interests. Each search consists set of $\{q_1,.,q_n\}$ search query components or dimensions (example like make of a car, model, cost etc.), where each search component is mapped to a tensor dimension M_n. The value η represents the number of searches made by a user. The value of η is variable as the number of searches made by different users are different. The value of σ represents the frequency of similar searches or searches which are similar in terms of searched dimension values for a user.

All other Greek symbols represent variable values, such as Υ represents highest scored or rated dimension values or recommendations, and β represents the ratings given to objects by users on a point scale of 1 to 5.

If a user has used a value for that search feature/parameter, the value is present in the vector otherwise, it is zero. Thus, for a user u_j the various searches/interests of a user can be represented as vectors as shown in eq. (7).

$$Iu_{j_\eta} = (q_1, q_2, ..., q_n)_\eta. \tag{7}$$

In case when $q_i = 0,$ or the searched parameter is not used/searched, the value of q_i noted as index in the tensor is represented by a zero value.

3.1 User Profiling

User profiling is one of the most important components of any Web personalized system. The following process chart represented by Figure 3.2 (processes as shown in continuation) below shows the methodology of creating various profiles in detail. Firstly, all available server log data is processed as per users and their searches. All object data (data about cars) is analysed and various unique parameters are identified.

In our case these parameters are the unique make, model, bodytype, search type and cost ranges. Once this is done a tensor model consisting of search data of users is created. Three types of tensor models catering to each category like group user profile, individual user profile and object profile can be created as shown in Figure 3.2. For group user profile all searches made by different users are considered. Similarly, for building an individual user's profile, a user's individual searches over a period of time are taken and a tensor model is built from it. For building object/item profile, detailed description of each objects are considered, eg. for building a car object model, properties like make, model, cost, car type (new, used, demo etc), bodytype and cost ranges are considered. These tensor models are then decomposed using various tensor decomposition techniques like PARAFAC, Tucker and HOSVD. To create group user profiles, clustering is done on the decomposed tensor of the respective users dimension in the model and once users are clustered in groups based on similarity of search behaviour, association rules are found out for searches of group members. Once such rules are established, *top* Υ unique rules are then stored in the group user profiles. Such *top* Υ rules are later used for making recommendations to a user. For creating individual user profiles top rated values in each dimension obtained after tensor decomposition are considered. This is similar to finding highest singular values from a matrix using the SVD. However, a major difference here is that instead of a two-dimensional matrix a tensor of higher dimension is taken. Similarly, to create object profiles, clustering is done on the decomposed tensor of the respective object identifier dimension (eg. dimension representing a car's identification number) in the model. Once objects are clustered based on the similarity of features, similar objects in a cluster are saved as values in the respective object profile.

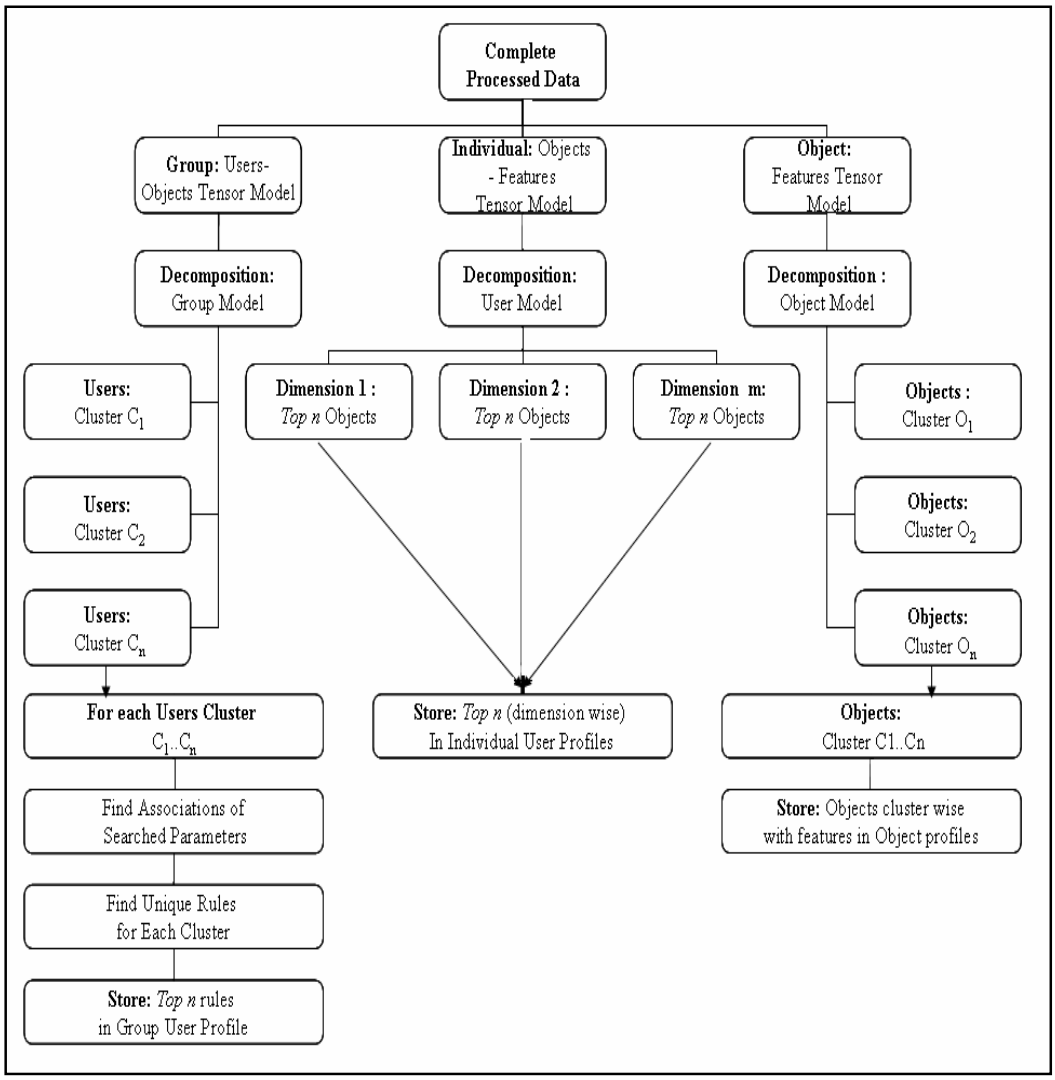

Figure 3.2 Complete detailed methodology adopted for making various types of profiles.

Identification of a user's intentions and interests is useful for any website to make effective recommendations. It is particularly true and relevant for every e-commerce website. This motive is crucial in determining the category of user profile to which a particular user may belong to. At any time, a visitor may have any one or a combination of these behavioural profiles as shown in Figure 3.3.

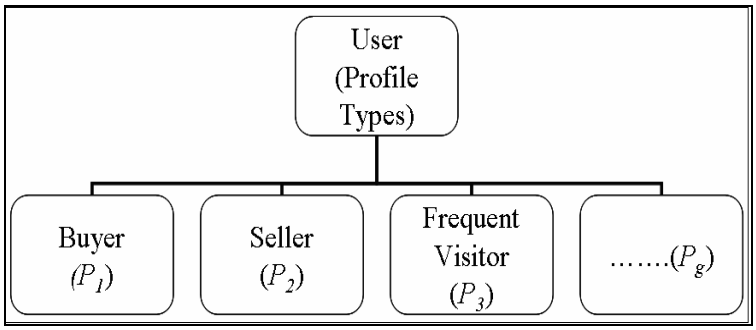

Figure 3.3 Various User Profiles identified.

Let $\{P_1, P_2, ..., P_g\}$ be the profiles categories as identified by a website and as shown in Figure 3.3. Thus for a user u_j his aggregate profile (P) may be any one of the identified profiles or a combination of two or more profiles and is denoted by $Pu_j = \{P_1 \cup P_2 \cup ... P_g\}$. To find the classification of a user profile category from the Web log data, statistical analysis of the data can be done. For identifying the specific category of a user's profile, searches made by users can be analysed. An example of how such categorization can be determined for '*Mileage Cars*' website is shown in Table 3.1. Here, searched keywords as given by users are extracted from the Web log files to deduce the category/categories of a user. In Table 3.1 a potential buyer is a user who is likely to buy a car and potential seller is a user interested to sell his car. Frequent visitor is a car enthusiast who makes large number of random, unmatched searches especially of new cars and browses through the technical details of most of them. On the other hand a casual visitor makes few searches.

S N	IP	Searched Keywords	Classified under category	Profile Category
1	A	"buy used cars"	Old Cars	Potential Buyer, Frequent Visitor
2	B	"wheels and tyres Sydney"	Servicing	Potential Buyer, Frequent Visitor
3	C	"used cars for sale in Gold Coast"	Old Cars	Potential Buyer, Frequent Visitor
4	D	"selling cars in NSW"	Old Cars	Potential Seller, Casual Visitor
5	E	"Selling my car"	Old Cars	Potential Seller, Casual Visitor

Table 3.1 Statistically identifying user profile types from data logs.

Data necessary to build users profiles can be stored and filtered in any desired manner. Typically user profiles data should be able to store and filter information based on the following data:

a) **Personal:** Age, sex, Occupation.
b) **Demographic:** Country, state, city/town, postcode, and distance from postcode.
c) **Social behaviour/Interests:** Searches, purchases, comments, rankings made by the users.

The data related to profile category is filtered and analysed to derive some knowledge from it. Preliminary analysis of the data can lead into valuable insights about the dimension values that have to be considered when building user or object models. It helps in selection of relevant values that are to be considered when building the user or object model. In case when the number of searched values in each dimension is large, only *top* Υ dimension values can be taken for modelling. Irrelevant dimension values that create unnecessary burden on computational resources can be discarded from the model.

3.1.1 Model Creation and Decomposition.

To create any tensor model, the data needs to be properly analysed and stored in a specific format. Let there be $U = \{u_1, u_2, ..u_z\}$ users. Since, two types of profiles such as individual user and group user profiles are proposed in this thesis, the data needed for creating the individual user and group users profiles needs to be processed separately to create the tensor models of each. The two most important steps that need to be taken on the processed data are

1) Model Construction (Building various tensor models from the processed data), and
2) Model Decomposition (finding top rated features and finding latent relationships between different features).

Once these two important steps are completed, clustering (Grouping similar user/objects) can be done. Since, this research work have modelled more than three dimensions for building each of the three categories of profiles such as individual, group and object, the visualization of the tensor model has been represented as shown in Figure 3.4.

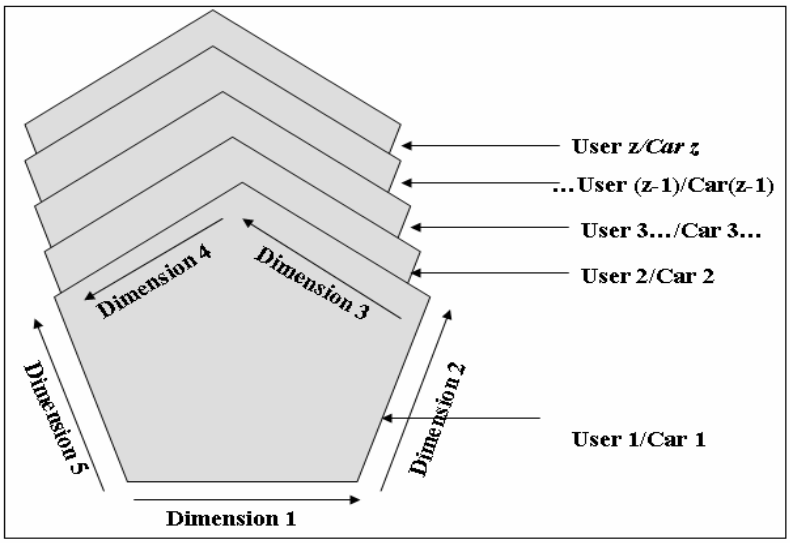

Figure 3.4 Visualization of a user tensor or an object (car) tensor in six dimensions.

3.1.2 Individual User Model Creation and Decomposition.

The major objective behind constructing individual user profiles is finding most relevant features in each dimension as searched by a user. Once, a user's interests are mapped and represented as a profile, they can be given higher preferences when making recommendations to such a user. The two steps undertaken for modelling an individual user's tensor are:

Step 1. Model Construction: For each user, the search data of all his sessions is analyzed. All unique features appearing in various user sessions are extracted to represent dimensions in the tensor model. A tensor is created with all such features. The user's tensor can be modelled as shown in Equation (8), where each $M_1, M_2 ... M_n$ represents the identified searched dimension compromising of the number of distinct search query features $(q_1, q_2, ..., q_n)$ provided by the website.

$$\mathcal{T}_z \in \mathbb{R}^{M_1 \times M_2 \times .. M_n} \tag{8}$$

Next, the term frequency of all similar interest vectors of a user u_j is found where $Iu_{j\eta}$ denotes the total interest vectors of a user. In case of an individual user u_j, two of his interest vectors are similar if $iu_{ji} = iu_{jk}$. The frequency value σ of such similar searches is counted. A similar search is a search where all searched dimension values are same. As an example, if for a car sales website a given user searches the same car

make, model, body type, search type and car cost in different queries and search sessions, then such interest vectors are similar. The frequency value of search vector iu_{ji} is saved as σ_{ji}. This interest vector when mapped with searched dimension values in the tensor model is saved as $(q_1, q_2, ..q_n) = \sigma_{ji}$. Thus, the frequency value σ_{ji} is the value of tensor at index position $(m_1, m_2, ..m_n)$ where $(m_1, m_2, ..m_n)$ is mapped to the values of searched query components $(q_1, q_2, ..q_n)$. All such unique interest vectors, representing a user's searches, along with the respective frequency value are fed into the tensor model. Individual user profile model construction is summarized in the algorithm in Figure 3.5. For each user the algorithm counts his distinct searches and inputs the index value mapped to the respective dimension of the tensor along with the user identification and frequency into the empty tensor $\mathcal{T}_z$.

Input: Processed Web log data with user searched parameters from $(q_1, .., q_n)$.

Let u_j be the user, with η number of interest vectors. The total interest vectors of a user are denoted as $Iu_{j\eta}$.

Output: Tensor $\mathcal{T} \in R^{M_1 \times M_2 \times M_3 \times .. M_n}$

Begin

1. *For* $k = 1..\eta$
   For $l = ((k+1)..\eta)$
   If $iu_{jk} = iu_{jl}$; //Count frequency of similar searches.
   $\sigma_{jk}++;$
   $Delete(iu_{jl});$
   Else
   $\sigma_{jk} = 1;$
   End if;
   End For;
   $iu_j^{\sigma} = \sigma_{jl}$; //Retrieve frequency of all distinct searches.
   End for;
2. $\{((q_1 ... q_n) = \sigma_{j1}), .., ((q_1 ... q_n) = \sigma_{j\eta})\}$; //Arrange grouped searches of user frequency wise.
3. Create an empty sparse tensor $\mathcal{T}_j$, and populate it with frequency $\sigma_{j\eta}$ and dimension values as.

$\mathcal{T}_j((q_1, ..., q_n) = \sigma_{j\eta});$

End

Figure 3.5 Algorithm for constructing Individual Users TSM from Web log data.

Step 2. Decomposition: In multi-dimensional data modelling, the decomposition process enables finding of the top rated dimension values (i.e. tensor entries and dimensions) as well as the hidden relationships that may exist between different dimensional values. As an example in a car sales website, the latent relationships between various car makes, models, costs can be ascertained by using the tensor decomposition, otherwise the relationships are difficult to analyze. These latent relationships could be answers to questions like how car makes and models are related and which different car makes are searched for and at what cost ?. To achieve tensor decomposition the thesis applies the popular and widely used PARAFAC [68], Tucker [177] and HOSVD [47] tensor decomposition techniques on the individual user models. These techniques have been discussed in detail in Section 2.2.2.4, but to reiterate how multi-dimensional decomposition is achieved, PARAFAC needs to be discussed briefly. PARAFAC is a generalization of PCA (Principal Component Analysis) to higher order arrays. Given a tensor of rank three as $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$, a R-component PARAFAC decomposed model can be represented as

$$t_{ijk} = \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr} + E \tag{9}$$

where a_{ir}, b_{ir}, c_{ir} are the i^{th} column of component matrices $\mathbf{A} \in \mathbb{R}^{I \times R}, \mathbf{B} \in \mathbb{R}^{J \times R}$ and $\mathbf{C} \in \mathbb{R}^{K \times R}$ respectively and $E \in \mathbb{R}^{I \times J \times K}$ is the three way array containing residuals. The value t_{ijk} represents an entry of a three-way array of $\mathcal{T}$ and in the i^{th} row, j^{th} column and k^{th} tube.

Thus, when the user's tensor (Equation (8)) is decomposed by PARAFAC [16], the various matrices formed are as shown in the Figure 3.6 below. In Figure 3.6, $\mathbf{M_1}, \mathbf{M_2} .. \mathbf{M_n}$ are the various component matrices formed after the decomposition of the tensor, where $m_{1(ir)}, m_{2(ir)}, .., m_{5(ir)}$ are the i^{th} column of component matrices $\mathbf{M}_1 \in \mathbb{R}^{I_1 \times R}, \mathbf{M}_2 \in \mathbb{R}^{I_2 \times R}, \mathbf{M}_3 \in \mathbb{R}^{I_3 \times R}, \mathbf{M}_4 \in \mathbb{R}^{I_4 \times R}$ and $\mathbf{M}_5 \in \mathbb{R}^{I_5 \times R}$ respectively (denoted using similar notations as used in Equation (9)), and $\lambda \in \mathbb{R}^{R \times R \times R \times R \times R}$ is the core tensor and the residual array/vector is the five way array containing residuals and denoted as $E \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4 \times I_5}$. In this thesis, R is the desired best rank tensor approximation for evaluating individual user profiles. The value of R is chosen from 1-6 in all the experiments. The best rank tensor approximation is the aggregate column-wise

representation of values in each dimension, where such values are represented in a matrix of size $\mathbf{M}_{i\times r}$. Higher-level rank decomposition creates matrices of large dimensions, thus flattening the tensor and defeating the overall purpose of dimension reduction.

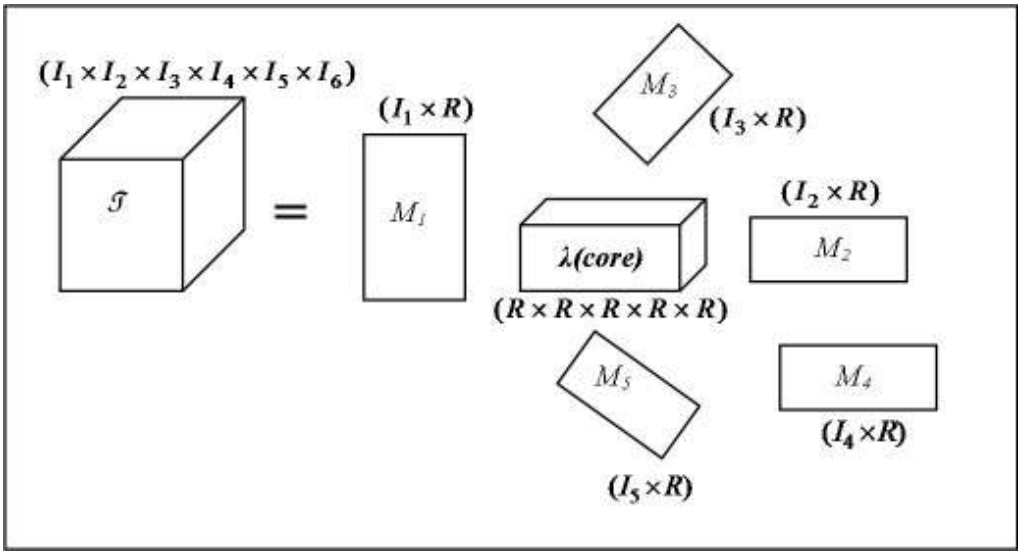

Figure 3.6 PARAFAC Decomposed tensor of Individual user's searches, gives these component matrices.

Once, individual user model construction and decomposition are done, the *top* ϒ values from each decomposed matrix, are saved in the individual user profile for making recommendations. The number ϒ that has to be saved in individual user profile depends upon the need of the Web service provider. This research work has chosen the *top* ϒ =15 values in each dimension to be saved in the user profile. These values are derived from decomposed matrices $\mathbf{M_1, M_2 .. M_n}$ (Figure 3.9) and these *top* ϒ highly relevant values are saved in the individual user profiles of each user. In the case where the tensor rank decomposition (denoted as R), $R = 1,$ such highest values can be learned out easily, but in the case where $R > 1,$ then an average of such row values is taken.

Even though individual user profile construction is an expensive method, it is a more suitable method to give the best recommendations to users because each individual user's personnel interests are taken into consideration. However, it is time consuming as each individual user's search data has to be processed and various models have to be created. To overcome this drawback, group user models can be created and used for making recommendations.

3.1.3 Group User Model Creation and Decomposition.

The major objective behind constructing group user profiles is finding users with similar search behaviour. Once similar users are determined, recommendations of similar objects within group members can be made. The other objective of creating group user profiles is helping to recommend objects to people who do not have an individual profile. This approach is similar to CF-based approach [73], [93], [99], [161], which recommends similar objects to users in a group, but unlike the CF-based approach in our case, recommendations can be made based on both the individual and group profiles of a user. Using both profiles ensures that a user would be recommended unique items based on the group profiles.

The method of creating group user profiles is similar to creating individual user profiles, but unlike individual profiles, here the searches made by a user are compared for similarity to each other. The other noteworthy aspect is that unlike previously used methods of evaluating similarity of user-items, the methodology adopted is a MDD approach for finding relationships between objects that are top rated and projecting latent relationships that exist between users and their searches. User to user comparison is done based on the total number of searches made by each user. The two steps undertaken for modelling group user's tensor are:

Step 1 Model Construction: Once, all processed data is available user-wise, session and search wise, it is analysed. All unique values of each dimension are derived to represent as dimension values into the tensor model. Each distinct index value represents a dimension value. A tensor is created with all searched feature values and the users are included in the model as last dimension. Thus, the overall structure of a group of users tensor consisting of n dimensions is shown in Equation (10), Section 3.1.3 where the dimension M_n represents the user's dimension.

$$\mathcal{T} \in \mathbb{R}^{M_1 \times M_2 \times ..M_n} \tag{10}$$

Group user modelling is summarized in an algorithm in Figure 3.7. The algorithm takes the processed search data of all users as input. For each user the term frequency of each similar search is counted. A similar search is a search in which all the searched parameters are same such as $iu_{ji} = iu_{jl}$. For certain query features a user may have not searched the feature, therefore in these cases when $q_i = 0$, an index value represented by a zero in the tensor for these dimension values has to be considered. The term frequency of all the searches of a user is determined. As an example for a user u_j , the term frequency of an interest vector iu_{jk}, denoted as iu_j^{σ} is σ_{jk}. This is entry value at the $m_1, m_2..m_{n-1}$ dimensions, where each $m_1, m_2..m_{n-1}$ represents the respective values at dimension $M_1, M_2...M_{n-1}$ of the tensor mapped to $(q_1, q_2, ..., q_n)$ query search components. The dimension value M_n holds all users details who have made the same search denoted as $(q_1, q_2, ..., q_n)$. Next TF-IDF (TV_i) values are determined (as shown in the algorithm in Figure 3.7) for all the searches of users, and the tensor is then populated with these values as $(q_1, q_2, ..., q_n, u_z) = TV_i$.

Input: Processed Web log data of all users with user id's and searched parameters from $(q_1,.,q_n)$.

Let $U=\{u_1,u_2...u_z\}$ be all users, where each user u_z has η interest vectors. Value of η varies for each user, as searches made by each user are different. The total interest vectors of a user are denoted as $Iu_{j\eta}$ or component wise as $Iu_j=(q_1,q_2,....,q_n)_\eta$. Let iu_j^σ be the frequency of k[th] interest vector, denoted as $iu_j^\sigma=\sigma_{jk}$.

Output: Tensor $\mathcal{T}\in R^{M_1\times M_2\times M_3\times..M_n}$

Begin

For each user in $u_z\in U$

1. //Count frequency of all distinct interest vectors of a user.

For $k=1..\eta$

For $l=((k+1)..\eta)$

If $iu_{jk}=iu_{jl}$; //Count frequency of similar searches.

$\sigma_{jk}++;$

$Delete(iu_{jl});$

Else

$\sigma_{jk}=1;$

End if;

End For;

$iu_j^\sigma=\sigma_{jk}$; //Retrieve frequency of all distinct searches.

End for;

2. Use $(tf*idf)$ to score these interest vectors. Where *TF*=Frequency of each distinct interest vector of a user (σ_{jk}), *ND*=Total number of interest vectors of a user (η) and *DF*=Number of similar interest vectors of all users.

$$TV_k=\sigma_{jk}\times\left(1+\log(\frac{\eta}{DF})\right).$$

$((q_1...q_n,u_j)=TV_k)$. // Save distinct Interest vectors with TF-IDF values.

End for

$(q_1...q_n,u_z)=TV_k$. //Arrange all grouped searches user wise, searched component and frequency wise.

3. Create an empty sparse tensor $\mathcal{T}$, and populate it with TV_k and dimension values of all users searches.

$\{\mathcal{T}(q_1,...,q_n,u_z)=TV_k\};$

End

Figure 3.7 Algorithm for constructing group users TSM from Web log data.

Thus, once the tensor has been populated with data, it can be decomposed by either PARAFAC, Tucker or HOSVD. In Figure 3.8, $\mathbf{M_1,M_2..M_n}$, are the various component matrices formed after the PARAFAC decomposition of the tensor, where $m_{1(ir)},m_{2(ir)},..,m_{6(ir)}$ are the i^{th} column of component matrices $\mathbf{M}_1\in\mathbb{R}^{I_1\times R}$, $\mathbf{M}_2\in\mathbb{R}^{I_2\times R}$, $\mathbf{M}_3\in\mathbb{R}^{I_3\times R}$, $\mathbf{M}_4\in\mathbb{R}^{I_4\times R}$, $\mathbf{M}_5\in\mathbb{R}^{I_5\times R}$, $\mathbf{M}_6\in\mathbb{R}^{I_6\times R}$ respectively and $\lambda\in\mathbb{R}^{R\times R\times R\times R\times R\times R}$ is the core tensor and *R* is the desired best rank tensor approximation, which is set as 1, 2, 3 and 4 in our experiments.

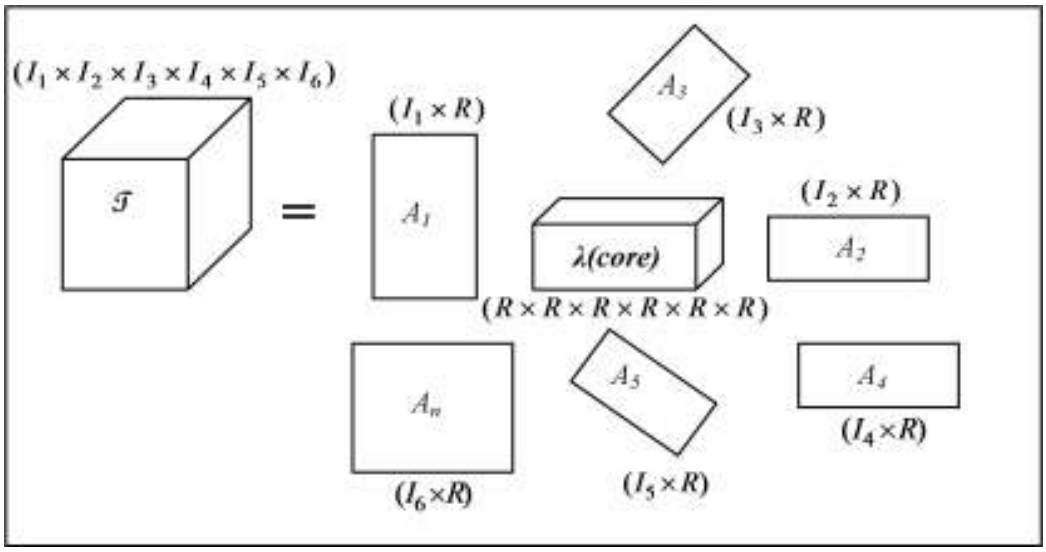

Figure 3.8 PARAFAC Decomposed tensor of group users searches, gives component matrices as shown.

Each of the component matrices $\mathbf{M_1}, \mathbf{M_2}..\mathbf{M_n}$, obtained after tensor decomposition represents the overall interaction of factors in that dimension. For example, if the dimension values in matrix $\mathbf{M_1}$ were taken**,** representing the car makes, all such car makes that have been searched highly by the users would be scored higher. Since, this research work has modelled the user as n^{th} dimension in all our tensor models, once decomposition is achieved the row values of matrix, $\mathbf{M_n}$ are taken as relevant values to be used for clustering. The values represented by matrix $\mathbf{M_n}$ shows users-users relationships based on the searches, each user has made. These values represent the aggregate relationships that exist between various dimensions of the tensor. In the case where the tensor rank decomposition (denoted as R), $R = 1$, the highest values can be learned easily, but in the case where when $R > 1$, then the average of such row values is taken. Once a group profile model is created and decomposed, the available information can then be used for making group user profiles.

3.1.4 Object Model Creation and Decomposition

An object model is a model based on the properties of objects/items that are being searched by users. Object model consists of item-item similarity based on the attributes or features of items. Two or more items may have similar characteristics. Take for example for an electronic E-commerce website. A digital camera with video

capabilities could be thought to be similar to a camcorder in terms of certain features. If camcorders are finished, users searching for such an item can be recommended alternative similar items.

The major objective behind creating object profiles is to compare object similarities based on properties of objects, so that similar search objects as searched can be recommended to users. The idea is to map a user's choice/search by recommending very similar objects. The other advantage of creating object profiles is since, an object model is built on static properties of objects, a new model only needs to be built when a new object is introduced. Creation of an object model may sometimes need domain experts or surveys to identify the important features or attributes that are relevant to an object. As an example in the case of creating a car object model, a domain expert or surveys can indicate the important attributes that attract a user's attention when buying a car. Once important attributes are identified, a tensor model can be created based on attributes representing the various dimensions of a user's search.

The two important methods for building object profiles using tensors discussed in this thesis include:

1. **Object models based on properties of objects:** An object model can be either built by finding similarity between different objects and their features (properties of objects)
2. **Object models based on ratings provided by users:** The other method to build object profiles is to utilize object ratings as provided by users.

Figure 3.9 details about the methodology of building object profiles, using either of the two methods.

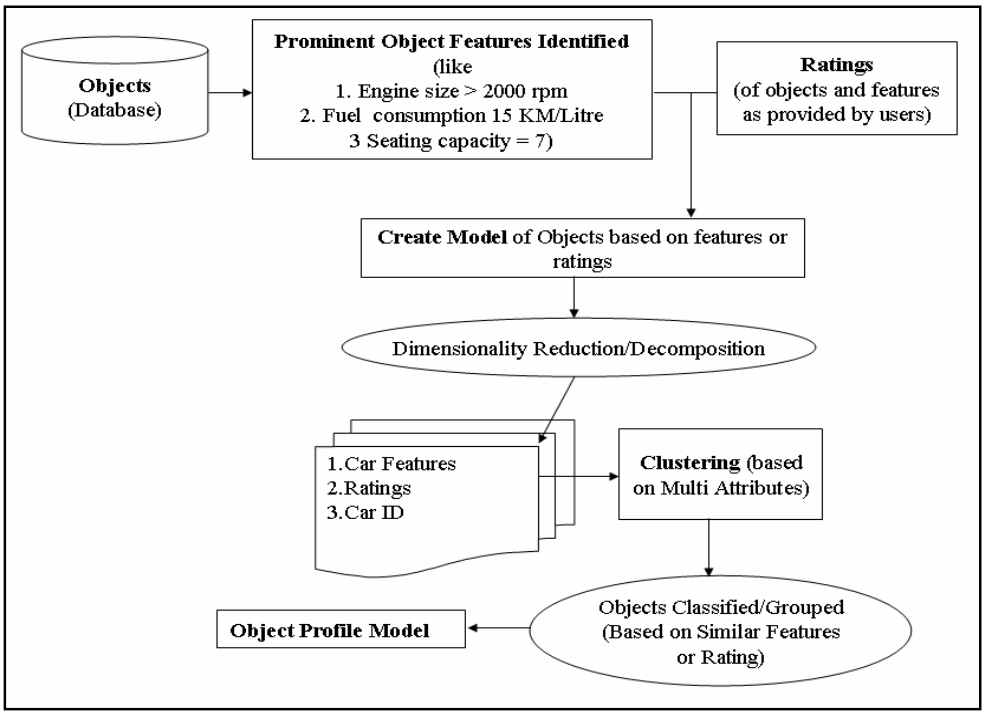

Figure 3.9 Complete methodology for building object profiles.

Thus, the whole process of object modelling consists of three main steps 1) Identification of dominant features or attributes of an object. 2) Model Construction. 3) Model Decomposition.

1) Identification of dominant features or attributes of objects needs careful analysis of the underlying website. Since, users have already been modelled based on object features that were searched in their individual and group profiles, therefore, to achieve better similarity between objects based on other features, that were not searched but were intrinsic part of object properties, certain features different from features that were searched were identified. These features are directly related to the granular details of objects such as in the '*Mileage Cars*' website where these features could be the number of kilometres done, transmission type, engine size, type of fuel used and the location of the car.

2) Model Construction: In case where no rating data is available, the object model can be created with features. Let there be $(q_1, q_2, ..., q_n)$ features based on which object model has to be created. Example for a car object model these features can be kilometres done, transmission, engine size, fuel used etc. Let there be o_z objects which have to be modelled. The vectors $(q_1, q_2, ..., q_n, o_z)$ represents values to be mapped in

the TSM. These values are mapped as dimension $M_1 \times M_2 \times ..M_n$, where each dimension M_{n-1} holds all distinct values for feature q_n. The structure of the object model based on features is shown in Equation (11). The object tensor $\mathcal{O}$ can be formed as

$$\mathcal{O} \in \mathbb{R}^{M_1 \times M_2 \times .. M_n}. \tag{11}$$

In the case where sufficient rating data is available, an object model can be built as shown in Equation (12). In this Equation (12), β represents ratings which are opinions given by users about certain features of an object. These ratings can be expressed on a point scale of say 1-5, where 1 is most preferred and 5 is least preferred. Let H be the number of object features of all objects O, which have been rated by users U. The tensor with three dimensions can then be created as shown in Equation (12), where each $\beta \times h \times o$ represents the rating value β on feature h of object o given by all users U. If similar ratings are given by multiple users, the frequency of such ratings is counted and stored as index value at $\beta \times h \times o$ position of the tensor.

$$\mathcal{O} \in \mathbb{R}^{\beta \times H \times O} \tag{12}$$

3) Model Decomposition: To decompose the object model either of PARAFAC, Tucker or HOSVD can be used. When the user's tensor (Equation (11)) is decomposed using PARAFAC by tensor toolbox [16], the various matrices formed are as shown in Figure 3.10. In the Figure 3.10, $\mathbf{M_1, M_2 .. M_n}$ are the various component matrices formed after the decomposition of the tensor, and R is the desired best rank tensor approximation. In the case where the TSM model is created using Equation (12), it will have three component matrices $\mathbf{M_1, M_2 .. M_3}$ instead of n components matrices (as shown in Figure 3.10) and the core matrix λ.

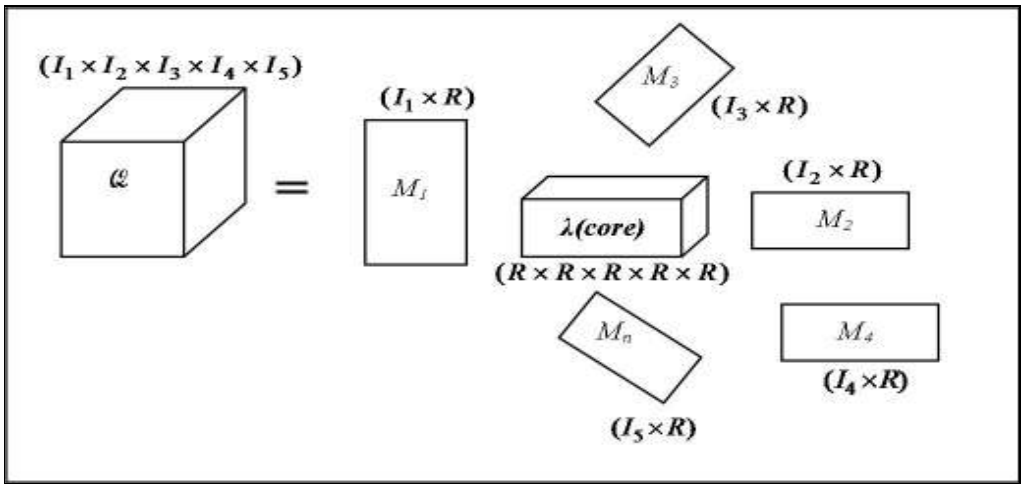

Figure 3.10 PARAFAC Decomposed tensor of Object Model, gives component matrices as shown.

Since, the car identification detail is modelled as the n^{th} dimension in all object tensor models, once decomposition is achieved the row values of matrix, $\mathbf{M_n}$ are taken as relevant values to be used for clustering. The values represented by matrix $\mathbf{M_n}$ depicts item-item relationships based on the different features of the objects. Whenever the tensor rank decomposition (denoted as R), $R = 1$, such highest values can be found out easily, but whenever $R > 1,$ then the average of such row values is taken.

3.1.5 Discussion: Model Creation and Decomposition

In this section, the various model construction techniques using tensors were discussed. Various model building strategies such as constructing models from an individual user's tensor; group of user's tensor; and object tensor consisting of top rated features as identified were discussed. Depending on a website's requirements any or all three models in combination can be used for making recommendations.

Model construction needs careful evaluation of all dimensions, as well as the number of values each dimension will hold. Model creation is highly dependant on the datasets and number of dimensions that are considered for modelling. Rarely used dimension values can create an unnecessary burden in terms of storage, efficiency and evaluation.

The values that are stored in the tensor are independent of the modelling style. No matter which type of tensor model is created, such as an individual user model, a group user model or an object model, the values are saved corresponding to the values that are mapped to the dimension index of the tensor. Since TF-IDF values are input as the values in the tensor, comparative search values, consisting of different searched dimensions and different users can simultaneously be co-related with each other. The other aspect which is independent of any model or method is the order of selecting dimensions when creating tensor models. To facilitate our modelling experiments users and objects have been taken as the last $\mathbf{M}_n$ dimension, but it is not mandatory to place dimensions in any particular order. The other consideration when choosing number of dimensions is that large number of dimensions flattens the tensor, thus increasing the tensor size unnecessarily. Therefore, it is better to choose small number of dimensions as the dimension size can be altered easily and instead of having multiple dimensions, certain similar dimensions can be aggregated as a single dimension.

Decomposition of each model can be achieved by using popular methods such as PARAFAC [68], Tucker [177] or HOSVD [47]. These three methods use different methodology for decomposition (see Section 2.2.2.4 for details), and most of the existing methods are derived from them. Using these three methods for decomposition would be sufficiently good enough to provide the best overview of the data and various relationships that exists between the users and objects.

One important consideration when decomposing tensor models is the choice of rank decomposition. Choosing best rank tensor decomposition needs careful evaluation. In this thesis, it was empirically found out that lower rank tensor decompositions are more accurate due to higher approximation of decomposed values. Tensor rank decompositions higher then the number of dimensions often perform poorly. Higher rank decompositions flatten the tensor, thus losing valuable information, hence, the values arising from it may not be appropriate for further analysis and modelling.

Once these models are created and decomposed, subsequently clustering can be applied to the various component matrices to find groups of similar users or objects. In the case of an individual user, the decomposed values of tensors in each dimension can be analysed for creating individual profiles. Clustering of such attributes is not needed for an individual user. When creating group user profiles, the decomposed

tensor values for the user dimension are clustered. From each cluster of similar users interesting patterns in the form of dimension values are saved in the group user profiles.

3.2 Clustering of Users and Objects

Web server log data containing usage patterns of users is a rich source of information about user's browsing habits. This information also assists in identifying the user's needs by inferring their intrinsic behaviour. A major problem for a website is to identify users with similar search behaviour. Clustering of users has been a popular approach for finding users with similar search behaviour and subsequently this information can be used for effective personalization and recommendation [128].

The major objective of conducting clustering on Web users search log data is to identify users who can be grouped based on similar search behaviour. All users clustered in a group can be recommended similar objects, which are popular in the group. On the other hand, the idea behind clustering different objects, based on their features is to identify similar objects, so that similar objects as searched by a user and objects that are similar to his search can be recommended. Some advantages and applicability of clustering users and objects are discussed below:

1) Similar users in a cluster can be recommended the most popular objects in the group such as the CF approach [73], [128], [135].
2) Fewer overheads in terms of system and resources, since group profiles would be created instead of individual user profiles.
3) New visitors can be recommended objects which are similar to their searches, as determined using the object models.
4) New and interesting objects, which may be quite different from the user's taste, can be recommended to a user.
5) Object similarity models can be used frequently with individual user profile models. Such models would be more effective, since a user would be recommended objects, which are more similar to his likings.
6) Object similarity models may effectively resolve drawbacks (new user, new item) of CF-based based methods dependent on ratings, as rating information may not be needed, since objects are clustered based on their features. Clustering of objects based on features or attributes may be helpful in reducing

problems faced by recommender systems that are purely dependant on rating information for making recommendations.

Clustering algorithms can be broadly classified into distance and density based methods [39]. In all the experiments conducted in this thesis a representative method consisting of both categories of clustering algorithms has been taken. To cluster users and objects we have considered two categories of clustering algorithms, the distance based clustering algorithms (like the k-means and advanced version of k-means namely the extended k-means) and the density or distribution based clustering algorithm depending on probability (like the EM (Expectation-Maximization) algorithm). The first category of clustering algorithms (k-means and extended k-means) are very popular, take less time for execution and in general show higher levels of accuracy when the data is processed, filtered and is represented as vectors. EM was taken so that a different measure apart from distance is also used to analyse and create clusters. This way, a different clustering approach can be tested and the difference in the quality of clustering given by two separate clustering approaches can be easily found out from the results. Similarly, clustering algorithms like direct, repeated bi-section and agglomerative have been used with the proposed FIBCLUS method to check the quality of clustering given by different approaches.

The WEKA implementations of k-means and its variants (extended k-means) [78], used in this thesis for experiments, do not have a procedure of automatically choosing optimal number of clusters. These implementations require the user to manually choose the required number of clusters apriori. However, in case of EM (Expectation-Maximization) method, the best number of clusters was chosen automatically by cross-validating the log-likelihood of the model's fit. The WEKA implementation of EM algorithm, as used in this thesis, automatically executes a 10 fold cross-validation and the log likelihood is averaged over all 10 results. In case of an increase in the log likelihood, a new clustering solution is generated and the 10 fold cross-validation begins afresh [78]. In this thesis, the 10 fold cross-validation clustering experiments were conducted with different seed values and 100 iterations of each. The log likelihood of each clustering solution was kept in consideration when choosing the best number of cluster n. The solution with the maximum likelihood estimate is chosen as the prefeered solution. The initial k value for k-means and its variant was

chosen as the value corresponding to the number of clusters in the best clustering solution given by the EM algorithm, *k=n* in this case.

3.2.1 VSM Clustering of Web Users Based on Search Logs

The processed Web log data of users represented as vectors or interests of each user is used to cluster Web users based on similarity of search behaviour. Searches made by each user on a website are processed for identifying search sessions. Each search session can be represented by a set of interest vectors for each user. For all users $U = \{u_1, u_2 ... u_z\}$, let each user u_j has $Iu_{j\eta}$ interest vectors. Each interest vector is consisting of $(q_1,.,q_n)$ search query features or dimensions. Thus $Iu_j = \{(q_1,q_2,...,q_n)_1,..(q_1,q_2,...,q_n)_\eta\}$, are the total interest vectors of user u_j. The frequency of similar searched feature values are found out from all the interest vectors of a user. Thus, from all the interest vectors of a user from $iu_{j1}, iu_{j2} ... iu_{j\eta}$, the frequency of similar searched feature values are evaluated. As an example, if a user has searched the car make 'Holden' three times then the frequency value of feature 'car make' is three. The whole process of the creation of a user's interests' vectors is explained in the algorithm presented in Figure 3.11. Once the processed data is ready, represented as weighted interest vectors of all users, the whole data is fed as input to the clustering algorithms (Expectation-Maximization and K-means).

Input: Processed Web log data of all users from $U = \{u_1, u_2 ... u_z\}$. Each user u_z has $1..\eta$ interest vectors, where each vector is denoted as $iu_j = (q_1, q_2, ..., q_n)$, consisting of $(q_1, ., q_n)$ search query components.
Output: Weighted Interest vectors of all users.
Begin
1. Create a matrix of dimension $\mathbf{A} = [a_{jk}]_{j=1...z;\ k=1..n}$; .
2. //Initialize user's interest vectors with 0 values.
   $(a_{jk}) = 0$;
3. For Each user $u_j \in U$
   For k=1 to *n*
   //Count frequency of similar searched components from all interest vectors of a user.
   $count(q_k)$, *where* $q_k \in Iu_j \wedge q_k \neq null$;
   $(a_{jk}) = \sigma_{zk}$; // For each user input frequency value σ_{zk} at respective index position of matrix.
   End For;
   End For;

$$\mathbf{A} = \begin{bmatrix} u_1\sigma_{11}\ u_1\sigma_{12}\ ...u_1\sigma_{1n} \\ u_2\sigma_{21}\ u_2\sigma_{22}\ ...u_2\sigma_{2n} \\ ..... \\ u_z\sigma_{z1}\ u_z\sigma_{z2}\ ...u_z\sigma_{zn} \end{bmatrix}$$ // Save matrix A.

4. Input matrix **A** for clustering.
End

Figure 3.11. Algorithm for Constructing Vector Model from Web Log Data.

3.2.1.1 VSM Clustering of Web Objects Based on Features

Since objects with their attributes details are available from the database, all attributes or features based on which objects have to clustered are identified. Once vectors of objects are created, they can be given as input to K-means, X-means (extended K-means) and EM (Expectation-Maximization) algorithms for clustering. The whole step of clustering objects is explained in the following steps.

Step1: Identify objects that have to be clustered based on attributes.

Step2: Retrieve features based on which clustering has to be done, in order to group objects.

Step3: Create vectors of such objects with relevant features, where each vector represents an object instance.

Step4: Cluster such object instances using K-means, X-means and EM clustering algorithms.

3.2.1.2 Limitations with VSM Clustering

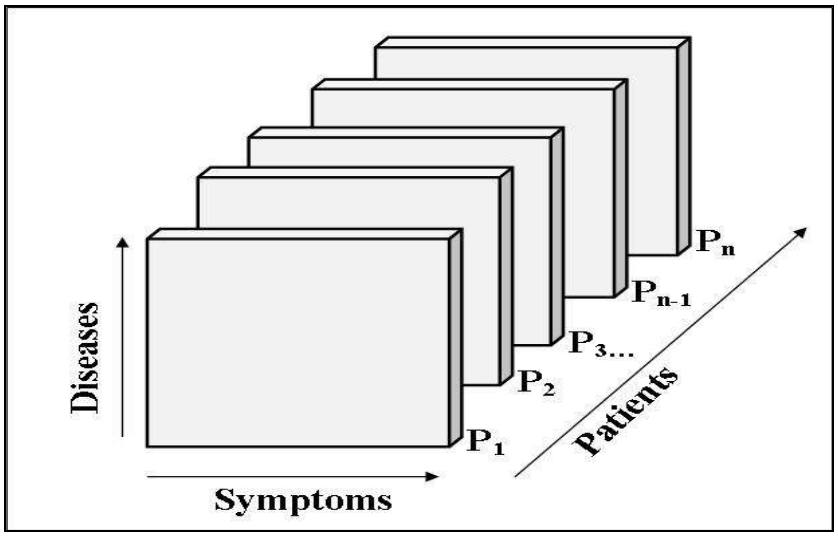

Figure 3.12 Tensor Representation of Patients/Disease/Symptoms Data.

Multi-dimensional datasets are common and handling such datasets needs specialised methods. Consider the following datasets consisting of one dataset related to a disease monitoring system and consisting of disease, symptoms and patients (Figure 3.12) or some other datasets which have already been applied to tensors to find the relationship of Web pages with the words, documents and links [97]; author, keyword and timestamp [52]; dataset for monitoring real time network traffic data (source-IP, destination-IP and port) [147]; or datasets from a chat room, consisting of users, keywords and time windows [2]. When using such MDD datasets, how is it possible to find patterns, relationships and latent concepts that underline the structure of such datasets? These patterns could be answers to questions such as which authors wrote similar books in a certain year, or which patients had similar disease symptoms or which users sent data to whom and through which port.

Consequently, deriving knowledge from such datasets using vector or matrix-based two-dimensional models may not be appropriate, as many valuable inter-component relationships may be lost. Thus, there is a need to obtain best clustering using other vector-based methods or using some high-dimensional data analysis methods so that the relationship between similar objects is maximised and good clusters of objects or users are obtained. The two proposed approaches are discussed below.

1. In this thesis, to handle object data with multiple characteristics, a Fibonacci based clustering method (FIBCLUS) [147] is proposed, which when applied to instance vectors of each object, has the ability to enhance clustering performance (Section 3.2.2).
2. The other proposed clustering method based on the MDD approach, using tensors is called as DIF (Dimensional Influence Factors). It is used to cluster similar Web users (Discussed in Section 3.2.3.1).

Both these approaches have been adopted to improve existing clustering solutions.

3.2.2 Proposed Fibonacci Based Clustering (FIBCLUS)

There are numerous examples where similar objects such as people and items have to be grouped based on similarity. The clustering process is an unsupervised method that systematically groups objects according to similarity shared amongst attributes. Evaluation of the similarity of attributes between instances is the core of any clustering method. The better a similarity function, the better the clustering results would be. Clustering sparse vectors with mixed attributes poses problems, as inter-cluster similarities may be too high to distinguish between a good clustering and bad clustering solution. If the dataset contains numeric attributes, distance measures such as Euclidean, Manhattan and cosine, are effective to evaluate the similarity between objects [77], [113].

However, when the dataset contains categorical (finite and unordered) attributes or a mix of numeric and categorical attributes then such distance measures may not give good clustering results [77]. Comparison of a categorical attribute in two objects would either yield 1 for similar values and 0 indicating that two instances are dissimilar. Based on the number of attributes matched these similarity measures evaluate how similar two instances are to each other. Such similarity measures are defined as overlap measure [31] and mostly suffer from the problem of clustering dissimilar instances together when the number of attributes matched is the same, but attributes that are matched are different [23]. Recently data driven similarity measures based on the frequency distribution of attributes have been a focus of research [23]. The proposed FIBCLUS method is a data driven approach, which clusters similar instances based on the characteristics of attribute values.

Datasets containing a mix of numerical and categorical attributes have become increasingly common in modern real applications, however, the existing similarity evaluation techniques are designed to fit each type of attributes separately [23], [77], [81] , [113], [141]. A general observation is that distance-based clustering techniques such as K-means, and the Expectation-Maximization work reasonably well with numeric data, and hierarchical algorithms work well with categorical data [77].

To cluster objects, a novel algorithm called as FIBCLUS (Fibonacci Based Clustering) that introduces effective similarity measures for numeric, categorical and a mix of both these types of attributes [147] is proposed. To achieve a constant separable distance between input attributes and to improve the clustering performance, the Fibonacci series [35] is used. A Fibonacci number is assigned to each attribute in order to alter the attribute values. For numeric data, each instance is given a global aggregate score relative to all other instances using the attributes modified with Fibonacci numbers. This global score is then used in clustering. Similarly, for categorical data, a global score is calculated for each instance with attributes modified with Fibonacci numbers. A pair-wise similarity matrix is then calculated using the number of attributes matched between each pair of instances and the assigned global scores. For the dataset with mix attributes, the global score of each instance is calculated according to the type of attributes. This pair-wise similarity matrix is then used in clustering.

3.2.2.1 Necessity for Clustering Methods with the Ability to Cluster Mix Attribute Datasets

Many techniques like average distance, mean distance or others can be used for evaluation of distances for numerical attributes based datasets. However, when pure categorical datasets or mixed datasets consisting of both the categorical and numerical attributes are to be clustered, the problem is how to measure the similarity between the instances represented by categorical attributes. Measures based on single link and complete link methods fail to give good clustering results as the single link method groups instances based on maximum similarity between any two pairs of clusters, and the complete link method clusters instances based on minimum similarity [141].

Let $X = \{X_1, X_2, .., X_b\}$ be a set of instances that are to be clustered. Each instance X_i is represented as $(x_{i1}, x_{i2}, ...x_{i\rho})$, where ρ is the number of attributes to be considered for clustering in each instance. The subscript in each single attribute $x_{i\rho}$ represents i the instance number and ρ as the attribute number. An attribute type can be either categorical or numerical. During the process of clustering, each instance is compared with other instances in order to find a similarity measure. One kind of similarity measure, called as overlap, between two instances X_i and X_j when the attributes are of categorical type is shown in Equation (13).

$$S(X_i, X_j) \equiv \sum_{k=1}^{\rho} \delta(x_{ik}, x_{jk}), \text{ where } \delta(x_{ik}, x_{jk}) = \begin{Bmatrix} 1, & x_{ik} = x_{jk} \\ 0, & x_{ik} \neq x_{jk} \end{Bmatrix}. \quad (13)$$

The use of such similarity measures (Equation (13)) may result in weak intra similarity when calculating the similarity between categorical attributes [113]. Other similarity measures for categorical attributes such as Eskin, Goodall, IOF, OF, Lin, Burnaby [23] are based on the overlap similarity measure and inherit the same problems. These measures attempt to reduce these problems by considering both the similarity and dissimilarity between instances, assigning 1 for a perfect match and arbitrary small values for a mismatch.

Moreover, in modern real-world applications, data with various instances containing a mix of both categorical and numerical attributes is common. In such cases the similarity measure described above may not provide correct values of instance similarity. A problem arises when assignment of an instance to a particular cluster is not easy. This problem is shown in the example of a deck of cards. A deck of cards typically outlines the problem of most clustering methods, which is that they are unable to cluster instances appropriately due to overlapping distances between them.

Consider a dataset containing a single deck of 52 cards and another dataset consisting of two decks of cards. Each deck of cards is identified by the distinct cover design it has. The data description of these datasets is detailed in Table 3.5. As the number of deck increases, the number of clusters and the complexity inherent within the clustering process increases. As the number of deck increases from $1..n$ the maximum number of perfect clusters increases to $4n$ where n is the number of decks.

For a single deck of cards the largest number of perfect clusters is 4, for two decks it is 8 and so on.

The ideal clustering results are shown in Table 3.3 for the deck of cards dataset problem. By ideal clustering, it means the number of clusters a person would group the cards based on all their attributes and the number of clusters desired. The corresponding clustering results obtained by different algorithms such as Expectation-Maximization (denoted as EM); K-means (KM); and extended K-means (XM), are shown in Table 3.4 and Table 3.5.

These were implemented in WEKA [78] with both Euclidian and Manhattan distances. Clustering using direct, repeated bi-section and agglomerative clustering techniques were used with both the cosine and correlation co-efficient similarity measures implemented in gcluto [141]. All clustering experiments using various methods as discussed (except with FIBCLUS) were run multiple times, and only the best results observed are reported for all the methods.

SN	Attribute Name	Attribute type	Value Range	Description
1	Card No	Numeric/ discrete	1-13	1-13 of all cards
2	Colour	Categorical	2	Red or Black
3	Category	Categorical	4	Hearts, Diamonds, Spade, Clubs
4	Deck Id	Numeric /Binary	1,2	1-1st Deck,2-2nd Deck

Table 3.2 Data description for deck of cards clustering problem.

2 Clusters	4 Clusters	8 Clusters
1-13, Red	1-13, Red, Hearts	1-13,Red , Hearts, D1
1-13, Black	1-13, Black, Spade	1-13,Red , Hearts, D2
	1-13, Black, Clubs	1-13,Red , Diamonds, D1
	1-13,Red, Diamonds	1-13,Red , Diamonds, D2
		1-13,Black , Spade, D1
		1-13, Black , Spade, D2
		1-13, Black , Clubs, D1
		1-13, Black , Clubs, D2

Table 3.3 Deck of cards cluster accuracy measure criteria (D1=deck1,D2=deck2).

SN	Clustering Algorithm	No of Cluster=2	No of Cluster=4
		Correctly Classified	Correctly Classified
1	EM	100%	100%
2	KM	100%	63.47%
3	XM	100%	73.08%
4	Direct	25%	38.5%
5	Repeated Bi-section	25%	48%
6	Agglomerative	48%	33%
7	Clustering Functions #1, #4,#5 #6 above with FIBCLUS	100%	100%

Table 3.4 Clustering results for a single deck of cards problem.

SN	Clustering Algorithm	Cluster=2 Correctly Classified	Cluster=4 Correctly Classified	Cluster=8 Correctly Classified
1	EM	100%	100%	48.07%
2	KM	98%	62.5%	56.7%
3	XM	98%	62.5%	56.7%
4	Direct	62.5%	36.5%	31.7%
5	Repeated Bi-section	65.5%	44.2%	31.8%
6	Agglomerative	65.5%	48%	25%
7	Clustering Functions #4, #5, #6 above with FIBCLUS	100%	100%	100%

Table 3.5 Clustering results for the two decks of cards problem.

Results in Table 3.4 and Table 3.5 clearly show that the stated clustering algorithms based on respective similarity measures performs satisfactorily with a single deck of cards, but as the complexity increases the clustering performance starts to decrease. This problem occurs due to the similarity methods adopted by such algorithms. Such methods are unable to handle the mix of attributes and their inherent relationships.

In all cases the similarity matrices obtained with the proposed FIBCLUS algorithm gives better results by using the existing clustering methods than by using the distance/density-based similarity evaluation algorithms alone. For all other cases, as the number of deck increases from one to two, the distance measures or similarity

methods employed by such methods start to overlap distances, resulting in weak intra-cluster similarity and clusters start to merge due to inseparable distances or higher inter-cluster similarity $S(C_1, C_1^c) \cong S(C_2, C_2^c)$ between the clusters. Here S is the similarity measure and C_1, C_2 are clusters with C_1^c, C_2^c as cluster representatives respectively.

Further the following Figures 3.13-3.16 illustrate the cluster assignments for the problem with the two decks of cards where only the best clustering results were taken from the original data as well as the Fibonacci altered data. The effect of using the Fibonacci series on the attribute values is evident in these figures. For the agglomerative algorithm, the best of the cosine and correlation co-efficient similarity measures were taken as a result. The equivalent results using FIBCLUS with the agglomerative clustering algorithm either with cosine or correlation co-efficient similarity functions are shown.

From Figure 3.13 it can be clearly deduced that cluster 1 and cluster 3 have overlapping distances, which consequently results in a weak clustering solution. In the case of Figure 3.15 the assignment of the same peak to a set of clusters shows the overlapping and, consequently a weak intra-cluster similarity value. However, it can be viewed from peaks in Figures 3.14 and 3.16 that with FIBCLUS the clusters were clearly identifiable. This directly points to the fact that the attribute values altered by Fibonacci numbers are able to create a good separable distance (weak inter-cluster similarity) between instances with the application of a distance function. The high peaks in Figures 3.14 and 3.16 which bind similar instances together confirm that the intra-cluster similarity was maximized, hence, resulting in the desired and optimal clustering for the underlying problem.

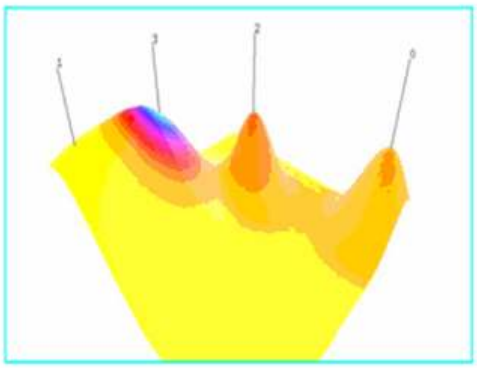

Figure 3.13 Agglomerative (4 Clusters).

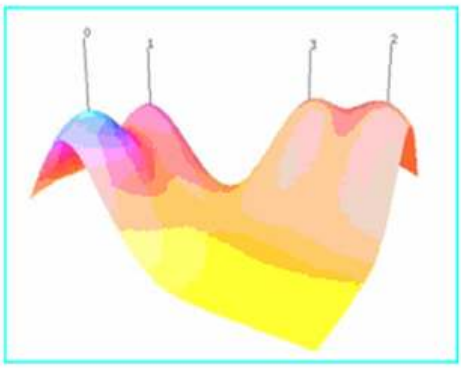

Figure 3.14 FIBCLUS with Agglomerative (4 Clusters).

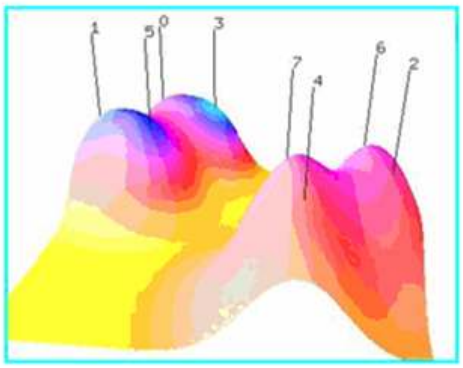

Figure 3.15 Agglomerative (8 Clusters).

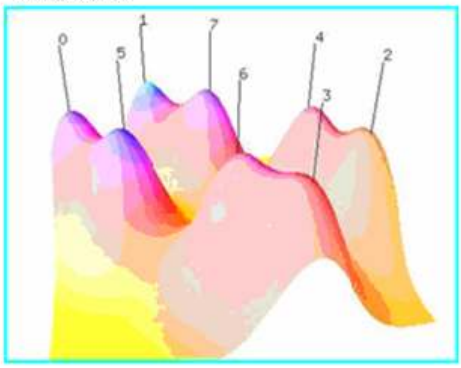

Figure 3.16 FIBCLUS with Agglomerative (8 Clusters).

The Fibonacci series maintains the constant ratio between the instances and creates a more separable distance between the instances. It helps in minimizing the inter-cluster similarity and maximizing the intra-cluster similarity as only the instances with similar attributes will be grouped together based on the transformation done by the Fibonacci series. It does so without altering the actual ratio between different instances, but maintaining the separation ratio φ between attributes.

3.2.2.2 Clustering Objects Based on the Proposed FIBCLUS Method

A brief detail about the Fibonacci is given below. Fibonacci series is discussed as the proposed clustering method FIBCLUS (Fibonacci based Clustering) uses the Fibonacci series to determine a global score for each instance and then utilizes the aggregate distance as a similarity function. The Fibonacci series is a sequence of numbers $\{F_l\}_{l=1}^{\infty}$ defined by the linear recurrence equation, $F_l = F_{l-1} + F_{l-2}$ with a fixed

value of the first two numbers $F_1 = 0$ and $F_2 = 1$. The first two Fibonacci numbers are 0 and 1, and each subsequent number is the sum of the previous two. An example of the first few numbers in the Fibonacci series is 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, etc. The Fibonacci series has been applied in many scientific and real life fields [35] from analysis of financial markets, to development of computer algorithms such as the Fibonacci search technique and the Fibonacci heap data structure [57].

Golden Ratio: One of the prominent properties of Fibonacci series is that the ratio of two successive numbers F_l / F_{l-1} tends towards 1.6 or φ, as l approaches infinity [35]. This value of φ is also known as the golden ratio.

Considering three neighbouring Fibonacci numbers $F_{(l)}, F_{(l+1)}$ *and* $F_{(l+2)}$, the ratio of $F_{(l+1)} / F_l$ and $F_{(l+2)} / F_{(l+1)}$ would be either the same or will tend to φ for larger values of l. Let us assume that this ratio is denoted as V_1 and is as shown in Equation (14).

$$V_1 = \frac{F_{(l+1)}}{F_{(l)}} = \frac{F_{(l+2)}}{F_{(l+1)}}. \tag{14}$$

Replacing $F_{(l+2)}$ by $F_{(l+2)} = F_{(l+1)} + F_l$, gives Equation (15).

$$V_1 = \frac{F_{(l+1)}}{F_{(l)}} = 1 + \frac{F_{(l)}}{F_{(l+1)}}. \tag{15}$$

Now from Equation (14), $\frac{F_{(l+1)}}{F_{(l)}} = V_1$, it becomes $\frac{F_{(l)}}{F_{(l+1)}} = \frac{1}{V_1}$, finally we have $V_1 = 1 + \frac{1}{V_1}$ or $V_1^2 = V_1 + 1$, which is in fact φ and represented with its relationship as $\varphi^2 = \varphi + 1$. The value of φ will tend to fall closer and closer to 1.6 as the ratio between two consecutive Fibonacci numbers is evaluated.

The primary purpose of using Fibonacci is that when categorical attributes are represented as equivalent numerical values and a distance-based evaluation method is used to cluster instances, the Fibonacci series acts as a constant separator between each attribute in all the instances. Since each attribute in all the instances is multiplied by a distinct successive Fibonacci number, only similar attributes in different instances will have the same values and will be clustered appropriately. Each

dissimilar attribute will be separable by a constant ratio of φ. If there are ρ categorical attributes in an instance which have been converted into equivalent numerical attributes then Fibonacci transformation of the attribute from $1...\rho$ will increase the ratio between $\frac{x_{i2}}{x_{i1}}, \frac{x_{i3}}{x_{i1}}, .. \frac{x_{i\rho}}{x_{i1}}$, however for two successive attributes it will always have a minimum values as φ.

Several similarity measures are proposed to deal with the respective attribute types of numeric, categorical, and mixed data [9], [23], [77], [81], [113], [120]. For numeric data, the popular similarity measures are Euclidean, Cosine and many others. Several weighting schemes such as normalization etc are used so that the various attributes containing different value ranges can be treated alike in measuring similarity. This is done so that the input data values of a clustering method are able to enhance the similarity measure in order to clearly identify the optimal clusters.

Most of the existing algorithms such as K-means are able to handle numeric data well, however the aim of using FIBCLUS with numeric data is to generate a search space in which the input instances are clearly distinguished. FIBCLUS represents each instance as an aggregate global value that represents all the attributes of that instance. In other words, if there are b numeric instances and ρ number of attributes then the FIBCLUS reduces the search space for each instance $X = \{X_1, X_2, .., X_b\}$ from ρ to 1 (eq. 16).

$$\mathbf{R}^b = \{(x_{b1}, x_{b2}..x_{b\rho})\} \rightarrow \{(x_{b1})\} \tag{16}$$

For categorical and mixed data the aim of FIBCLUS is to identify the best possible similarity that exists between a pair of instances by considering all the attributes. The score of all attributes in this case is also represented as an aggregate global score, but, unlike the overlap and similar measures [31] where similarity between instances is computed without any consideration for numeric data, FIBCLUS enhances the similarity by comparing the numeric attributes and reflecting its effect on the overall similarity measure.

There are three cases, which have to be considered when clustering with FIBCLUS:

1) When the dataset contains only numeric attributes.

2) When the dataset contains only categorical attributes.

3) When the data is a mix of both numeric and categorical attributes. The clustering process for each case is discussed separately.

Given the set of instances $X = \{X_1, X_2, .., X_b\}$ with ρ number of attributes $(x_{i1}, x_{i2}, ...x_{i\rho})$, a Fibonacci number is initialized for each attribute maintaining the golden ratio φ. Let $F = \{F_1, F_2 ... F_\rho\}$ be the set of Fibonacci numbers chosen corresponding to ρ the number of attributes, where each successive Fibonacci number $F_{\rho+1}$ maintains the golden ratio φ with the preceding number F_ρ. In the experiments F_1 is initialized as $F_1 = 5$ because the series starts to get closer and closer to φ from here onwards. Consider the example of a dataset of four attributes $x_{i1}, x_{i2}, x_{i3}, x_{i4}$, where $F = \{5, 8, 13, 21\}$ is the set of Fibonacci numbers. In this case $F_1 = 5$ is used to transform x_{i1} and $F_2 = 8$ is used to transform x_{i2} and so on. A value in F_ρ maintains the golden ratio as $F_2 / F_1, F_3 / F_2, F_4 / F_3 \cong 1.6$.

Case 1: This case applies to the datasets that have instances, which have only numeric attributes. The first step is finding the maximum value of each attribute in all the instances. Let this value be denoted as $\max(x_1), \max(x_2), ... \max(x_\rho)$. The maximum value of each attribute is used for normalizing the attribute values. Normalization is done to scale the values in a constant range so that the Fibonacci number chosen for that attribute does not drastically change the golden ratio φ, which separates the values of one attribute from another. Since normalization would give 1 as the maximum value, multiplication with a Fibonacci number would distinctly separate all such values from others and it would maintain the golden ratio between different attributes. Thus, all the attributes of each instance X_i are normalized as shown in Equation (17).

$$x_{i\rho} = \frac{x_{i\rho}}{\max(x_\rho)}. \tag{17}$$

Next, the instance score is evaluated as shown in Equation (18).

$$Score(X_i) = \sum_{j=1}^{\rho} \frac{x_{ij}}{\max(x_j)} \times F_j. \tag{18}$$

Once each instance is mapped to an aggregate global score, a distance-based clustering algorithm such as K-means (KM) or Extended K-means (XM) or a density-

based algorithm such as Expectation-Maximization (EM) can be used to group this reduced complexity data $\mathbf{R}^b = \{(x_{b1}, x_{b2}..x_{b\rho})\} \rightarrow \{(x_{b1})\}$.

Case 2: This case applies to the datasets that have instances, which have only categorical attributes. All categorical instances are mapped into numeric values. Each instance X_i with attributes as $(x_{i1}, x_{i2}, ...x_{i\rho})$, is assigned a score using the Fibonacci series as shown in Equation (19).

$$Score(X_i) = \sum_{j=1}^{\rho} x_{ij} \times F_j. \tag{19}$$

The number of common attributes between two instances is denoted as $X_i \cap X_j$. Each instance is compared for similarity with other instances. The pair wise instance similarity is then calculated as shown in Equation (20).

$$Similarity(X_i, X_j) = \frac{X_i \cap X_j}{\rho} + \frac{Score(X_i)}{Score(X_j)} \tag{20}$$

where $Score(X_i) <= Score(X_j)$, this condition ensures that the similarity calculation is done only once for a pair (x_{i1}, x_{j1}). The pair-wise similarity matrix becomes the input to a clustering algorithm.

Case 3: This case applies to the datasets that have instances with both numeric and categorical attributes. Let k be the number of categorical attributes and l be the number of numeric attributes, where $k + l = \rho$. All categorical attributes are mapped into numeric representation and for the numeric attributes, the highest value of each attribute denoted as $\max(x_1), \max(x_2)..., \max(x_l)$ from all instances is found. The score of each instance is determined separately based on the values of both numeric and categorical attributes. For instance X_i with attributes as $(x_{i1}, x_{i2}, ...x_{i\rho})$, it is calculated as shown in Equation (21).

$$Score(X_i) = \sum_{1}^{k} (x_{ik} \times F_k) + \sum_{k+1}^{\rho} \frac{(x_{il})}{\max(x_l)} \times F_l. \tag{21}$$

The instance similarity between two instances X_i, X_j is based on Equation (21) above, and is evaluated as shown in Equation (22).

$$Similarity(X_i, X_j) = \frac{X_i \cap X_j}{\rho} + \frac{Score(X_i)}{Score(X_j)} \quad (22)$$

Where $Score(X_i) <= Score(X_j)$.

The pair wise similarity can be calculated for each pair of instances. Let the similarity between two instances X_i and X_j be denoted as $Similarity(X_i, X_j)$**.** This similarity is nothing but the global aggregate score (a_{ij}) between the instances. Thus, the overall similarity score between all instances is represented by matrix $\mathbf{A} = [a_{ij}]_{b \times b}$**,** which becomes the input to the clustering algorithm. The complete FIBCLUS algorithm is explained in Figure 3.17. At first successive Fibonacci numbers corresponding to the number of attributes in an instance are initialized starting with $F_1 = 5$. For each categorical attribute $x_{i\rho}$ convert all the value into distinct numerical values. As an example, if for the car attribute model name, if four models such as (Commodore Rav4, Matiz and Civic are present then they can be mapped into numeric values 1, 2, 3, 4 respectively. Next, all instances from $X = \{X_1, X_2, .., X_b\}$ each consisting of $(x_{i1}, x_{i2}, ...x_{i\rho})$ attributes is altered by the respective Fibonacci number mapped with the attribute values. Both numeric and categorical attributes are scored using different schemes, and finally global score per instance is evaluated.

Input:

$X = \{X_1, X_2, .., X_b\}$; // Datasets instances with ρ attributes as $x_{i1}, x_{i2}, ..., x_{i\rho}$.

$F = \{F_1, F_2 .. F_\rho\}$. //Successive Fibonacci numbers F corresponding to each $1..\rho$ attributes.

$F_j x_{ij}$ = Categorical attribute values, mapped into numeric value.

Output:

$\mathbb{R}^b = \{(x_{b1})\}$; // Numeric instances: Global score

$\mathbf{A}_{b\times b}$; // Instances with Categorical or Mixed attributes: Similarity Matrix.

Begin:

Step 1. $F_1 = 5$; // Initialize Fibonacci series.

$\mathbf{A} = [a_{ij}]_{i=1...b, j=1..b}$; //Create similarity matrix.

$(a_{ij}) = 0$; //Initialize similarity matrix **A.**

Step 2. // For numeric attribute $\max(x_{i\rho})$ finds maximum attribute value from instances.

For each j=1 to ρ;

$\max(x_j)$

Step 3. // Evaluate scores for each instance.

For each i=1 to b;

$Score(X_i) = 0.0$;

For each j=1 to ρ;

If domain (x_{ij}) = Numeric

$$Score(X_i) = Score(X_i) + \frac{x_{ij}}{\max(x_i)} \times F_j.$$

Else domain (x_{ij}) = Categorical

$Score(X_i) = Score(X_i) + F_j x_{ij}$.

//Similarity between attributes is reduced to $\mathbb{R}^b = \{(x_{b1})\}$.

Step 4. //Calculate similarity between instances with mixed attributes or purely categorical //attributes.

If domain (x_{ij}) = Categorical

For each i=1..b;

For each j=1..b;

If ($(Score(X_i) <= Score(X_j))$

$$Similarity(X_i, X_j) = \frac{X_i \cap X_j}{\rho} + \frac{Score(X_i)}{Score(X_j)};$$

$(a_{ij}) = Similarity(X_i, X_j)$

$\mathbf{A} = [a_{ij}]_{b\times b}$;

Return **A** ;

End.

Figure 3.17 Complete FIBCLUS Algorithm.

3.2.2.3 FIBCLUS Clustering of Objects Based on Features

Objects with various attributes or dimensions can be used to cluster very similar objects using FIBCLUS. Let there be $\{o_1, o_2, ...o_z\} \in O$ objects that have to be clustered. Let $x_{z1}, x_{z2}, ..x_{zn}$ be the various attributes or features based on which such objects have to be clustered. The complete method of clustering objects is

summarized in the steps as below. Let F_n be the number of Fibonacci numbers chosen corresponding to *n,* the number of attributes.

1. Initialise Fibonacci numbers $F_{1..n}$ corresponding to number of attributes where $F_1 = 5$.
2. Convert each categorical attribute into distinct numerical values.
3. Once all categorical attributes are concerted into distinct numerical values, multiply each attribute $x_{z1}, x_{z2}, ..x_{zn}$ with the respective successive Fibonacci number. Such as $x_{z1} \times F_1, x_{z2} \times F_2, ..x_{zn} \times F_n$.
4. Normalize the scores of each object o_z attribute wise as $\frac{score(x_{zi})}{\max(x_i)}$.
5. Compute score of numerical attributes and categorical attributes using Algorithm in Figure 3.17
6. Once global score are evaluated, cluster the scores using K-means, EM or any hierarchical clustering method.

3.2.2.4 Discussion: Clustering with FIBCLUS

The problem with clustering the datasets with categorical and mix attributes (having both numerical and categorical attributes) is that, comparing individual instances in order to find similarity and dissimilarity for each attribute value is an expensive method [23]. The complexity of this process can be reduced by mapping all instance values (independent of what attribute domain they are in) to an aggregate score considering all attributes and, then, any clustering algorithm could be applied onto this reduced search map to cluster these instances.

The proposed FIBCLUS method showed that it has the potential to represent an instance by a single aggregate value and then perform clustering based on similarity matrix. However there is an extra overhead in terms of time and space due to the additional step of calculating the global scores and similarity scores, however, it is compensated by the reduced search map during the clustering process. Moreover, clustering usually is an offline process and it is affected more by accuracy than the time and space complexity.

Since, Web log data may contain multiple instances or multiple searches of the same user, under these circumstances it becomes inefficient to use FIBCLUS to cluster such

data. However, FIBCLUS can be used to cluster objects based on features. Since each single object instance represents a complete object with many parameters, clustering on such data can be performed to find near similar objects.

3.2.3 Tensor Clustering of Web Users Based on Search Logs

Clustering is done on the component matrix $\mathbf{M_n}$ obtained after tensor decomposition methods, and representing decomposed values of the last dimension M_n. Each component matrix from $\mathbf{M}_1, \mathbf{M}_2, .., \mathbf{M_n}$ is of dimension $\mathbf{M}_{i \times r}$ where i is the number of ways in a dimension M_n, and r is the value of best rank approximation of the tensor. Since, the last component matrix $\mathbf{M}_n$ represents the users dimension in the proposed data model, this matrix becomes the input to the clustering process. The entries (or values) in the matrix $\mathbf{M}_n$ depicts the correlations of each user with other users based on the multiple factors of the tensor obtained after the interaction of various factors/dimensions (Figure 3.8). Therefore, clustering on the row values of matrix $\mathbf{M}_n$ results in grouping of users according to similar search behaviour, while keeping the various searched dimensions in consideration. Clustering on rows of tensor decomposed matrix $\mathbf{M}_n$ of users dimension is achieved by using three clustering methods, these are EM (Expectation-Maximization), K-means and X-Means [78]. This research work has used three different methods to evaluate the effectiveness of distance (K-means, X-Means) and density-based (EM) clustering algorithms when used for clustering tensor decomposed matrices.

3.2.3.1 Tensor Clustering of Web Users Based on the Proposed DIF method

This section discusses the proposed DIF (Dimensional Influence Factor) method that can enhance clustering of tensor decomposed objects or users in a multi-dimensional scenario. The sole objective of using DIF as an extra input during clustering methods is to minimize the information loss, which may have occurred during the decomposition process of the tensor approximating the higher order data to a lower order dimensional data. The dimensional influence factor (DIF) is the factor which defines how each dimension influences a user's search. It is evaluated individually for

each dimension but combined linearly to collectively represent a user's overall search behaviour.

Once, group users model is created and decomposed, clustering is done to find similar users. To achieve the clustering, the matrix values from $\mathbf{M_1, M_2 ... M_{n-1}}$ are integrated for each user to get one DIF value. These DIF values for each user are then given along the decomposed values of matrix $\mathbf{M_n}$ of each user to the clustering algorithm. The complete method of evaluating DIF is explained below.

The ideal situation when a user has searched all search feature values in a dimension is DIF with a value equal to 1. Let iu_z^n be a user's interest in a dimension M_n and μ_n be the sum of component matrix of best rank approximation for this dimension. Consider a tensor $\mathcal{T} \in \mathbb{R}^{M_1 \times M_2 \times M_3 \times .. M_n}$ of users, $\{u_1, u_2, .. u_z\} \in U$. Each dimension $M_1, M_2 ... M_{n-1}$ has a fixed number of values or ways in it. For example, let dimension M_3 represent car body types and have seven different car body types. A user search for body type (such as the hatch, sedan, utility, van, luxury, SUV or None) can include any one or none of these body type values. Let the number of distinct search feature values searched by a user in each dimension be represented as $q_1, q_2 ... q_n$. Let for the body type search query feature the user u_j has searched distinctly for two car body types out of total seven. (E.g. user u_j searched for hatch and sedan, thus in this case q_3=2). Thus a user u_j's interest iu_j^n in a dimension is shown in Equation (23).

$$iu_j^n = \frac{q_n}{|M_n|} \tag{23}$$

The component matrices of tensor best rank approximations from $\mathbf{M_1, M_2 ... M_{n-1}}$ obtained after PARAFAC, Tucker and HOSVD decompositions, and representing decomposed values of each dimension $M_1, M_2, ... M_{n-1}$ except the last dimension are taken. Each best rank approximation is a matrix of dimension $\mathbf{M}_{i \times r}$ where $i = 1..n$, and r is the value of desired best rank approximation of the tensor. Each dimension may hold a certain importance for each user. Some dimension values may be highly searched by users, whereas some dimension values may not be searched by user at all. Tensor decomposition is able to evaluate such top rated interest values for each

dimension, but, it does not include aggregate values, rather, it represents such values in the last component matrix $\mathbf{M_n}$, depicting users overall search behaviour.

Therefore, an aggregate value representing all dimensions except the user's dimension has to be evaluated. Equation (24) represents such an aggregate value of tensor decomposition that reflects overall search behaviour in all searched dimensions. For each dimension from $M_1, M_2, ...M_{n-1}$ value of μ_n (sum of component matrix in a dimension) is evaluated row-wise denoted as $m_{i\times r}$ and averaged by dividing with the best rank approximation value (r) of the tensor. For each matrix from $\mathbf{M_1, M_2 ... M_{n-1}}$, where n is the number of dimensions, it is evaluated as:

$$\mu_{n=} \frac{\sum m_{i\times r}}{r} \tag{24}$$

The value of DIF is then calculated linearly for each user u_z as shown in Equation (25). Thus, it combines a user's interest in a dimension with the aggregate tensor decomposed values in each dimension and gives an aggregate factor by which the tensor decomposition may have lost some information (in matrix $\mathbf{M_n}$) about user's personal interests. The DIF value is then used with matrix $\mathbf{M_n}$, for clustering user's with similar search behaviour.

$$DIF_{(u_z)} = \sum_{1}^{n} \frac{\mu_n}{iu_j^n} \tag{25}$$

Consider the example of calculating DIF for the tensor model $\mathcal{T} \in \mathbb{R}^{50\times 60\times 70\times 10}$ modelling 10 users with 4 dimensions as $\mathcal{T} \in \mathbb{R}^{Make\times Model\times Bodytype\times User}$. Let the sum of component matrices except the last dimension of PARAFAC decomposition with best rank 1 gives values of μ in each dimension from $\mathbf{M}_1 ... \mathbf{M}_3$ except the last dimension '*user*' (users-users relationships represented by matrix $\mathbf{M}_4$) as $\mu_1=0.025$, $\mu_2=0.075$, and $\mu_3=0.055$. If a user u_j made 3, 12, 10 distinct searches in each of the three dimensions respectively, then the aggregate value of his interest in each of the three dimensions is

$iu_j^1 = \frac{3}{50}, iu_j^2 = \frac{12}{60}, iu_j^3 = \frac{10}{70}$, Therefore $DIF = \frac{0.025}{3/150} + \frac{0.075}{12/60} + \frac{0.055}{10/70} = 0.1846$

If for a dimension the value of iu_j^n is equal to 1, it signifies the flexibility of the search behaviour of a user in the dimension. Value of $iu_j^n = 1$ means a user has searched all the available values in this dimension. When personalizing the interest of such users for making recommendations in this dimension, a wider range of categories can be included. On the other hand, a value of iu_j^n close to 0 indicates that the search is rigid, which means, user only searches a few selective object values for this dimension. Thus, a small value of iu_j^n signifies a user's interest in this dimension. When enhancing recommendations for this dimension, utmost care should be taken, so that only dominant values are recommended rather then recommending too many objects. On the other hand if this value is equal to 0 it means that the user is not interested in this dimension or has not searched even a single attribute value in this dimension. This relationship is expressed by the following Equation (26) of a user's aggregate interest iu_j^n in each dimension.

$$iu_j^n = \begin{pmatrix} 0 = \textit{No Search, Un-interested in search dimension, If } q_n = 0. \\ iu_j^n > 0 \wedge iu_j^n < 1 = \textit{Rigidity in search behaviour; If } q_n / |M_n| > 0 \\ 1 = \textit{Searched all dimension values, Flexibilty, If } q_n = |M_n|. \end{pmatrix}; \quad (26)$$

3.2.3.2 Tensor Clustering of Web Objects Based on Features

Once an object model is created and decomposed (as discussed in Section 3.1.4) similar objects can be clustered based on features. The group of objects, which are clustered in the same cluster, are objects that have many similar features. To cluster similar objects, the items-items similarity measures using vector methods have adopted Euclidian, Manhattan and cosine similarity measures. For evaluating the items-items similarity measures, tensor-based methods use decomposition matrices obtained from the respective tensor decomposition technique. To cluster similar objects the last component matrix $\mathbf{M}_n$ is taken, which represents the object's identification details in the proposed data model (Figure 3.10). This matrix becomes the input to the clustering process, as entries (or values) in the matrix $\mathbf{M}_n$ depicts the correlations of each object with other objects based on the multiple relationships that exists between all dimension or features of the objects.

Finally, clustering on the values of matrix $\mathbf{M}_n$, results in grouping similar objects based on their features. Such groups of objects can be used for making recommendations to an anonymous user, who does not have any profile information.

3.2.3.3 Limitations with Tensor Clustering

Tensor clustering inherits the problems faced by tensor models due to adaptation of tensor decomposition components in clustering. Web data is complex, and effective identification of dimensions is an important factor for effective modelling of users behaviour. Some drawbacks of tensors and inhereted clustering methods are:

1. Due to the very large volumes of data and dimensions, tensor models built on such datasets may have memory and space complexity issues On the other hand in the cases where there are large numbers of dimensions, but less data, data sparsity is a common problem with tensors [96], [98]. Memory and space problems can arise during model building or when decomposing or doing dimension reduction.
2. Unlike matix based methods like SVD, tensor decomposition methods are not unique. Most of the existing new methods discussed are variations of the three most widely used tensor decomposition techniques like the Tucker, PARAFAC and HOSVD.
3. The other major issue with tensors is choosing the best rank decompositions for each data model. Lower rank tensor decompositions perform much better due to higher approximations. On the other hand if higher rank decomposition values are chosen for clustering then compared objects may loose valuable information and the performance of such methods may be equal to or be worse than the two-dimensional methods.
4. Since Web users data is dynamic, to update a user profile, such models need to find ways of incremental updates rather than updating entire profiles.

To resolve these issues some proposed solutions are

1. To tackle the problem of memory and space, user's can be identified based on the frequency of visits or on the category of visitor like buyer, seller etc (Figure 3.3, Section 3.1). Separate models can then be created for each category of users. The other resolution to such problems is by choosing the

appropriate dimensions (Section 3.1), all dimensions and dimension values that are unimportant can be removed from the model. The other solution to such problems is that with the introduction of new methods of calculating tensor decompostions like Memory Efficient Tucker, MET [98], MACH [146] such issues can be minimized.

2. Since the three most popular tensor decomposition methods like PARAFAC [68], Tucker [177] and HOSVD [47] each adopt different approaches, the best solution would be to use all three in conjunction to obtain a better overview of the decomposition and modelling of users and objects.
3. Finding the optimal rank for decomposition is a big challenge, however this research proposes to use decomposition upto the rank before which clustering performance starts to decline. To evaluate so, a detailed empirical evaluation of all clustering using various tensor ranks is performed.
4. Tensor models have the ability to adapt and incorporate a user's new searches/interests. Whenever processed data of new searches is updated, only such data can be incrementally added to the model, and later once the new data is populated in the tensor model, a single decomposition will help in finding the interest drift in a user's search behaviour. The other method to avoid using regular incremental changes is to use information in the object profiles for making recommendations. The advantage with object profiles is that new objects can be introduced without much complexity.

3.3 Profile Creation Methods

This section of the thesis discusses about various profile construction techniques for an individual user, group user and object profiles. The complete methodology adopted for making profiles for individual users group users and objects profile is explained in the Figure 3.18.

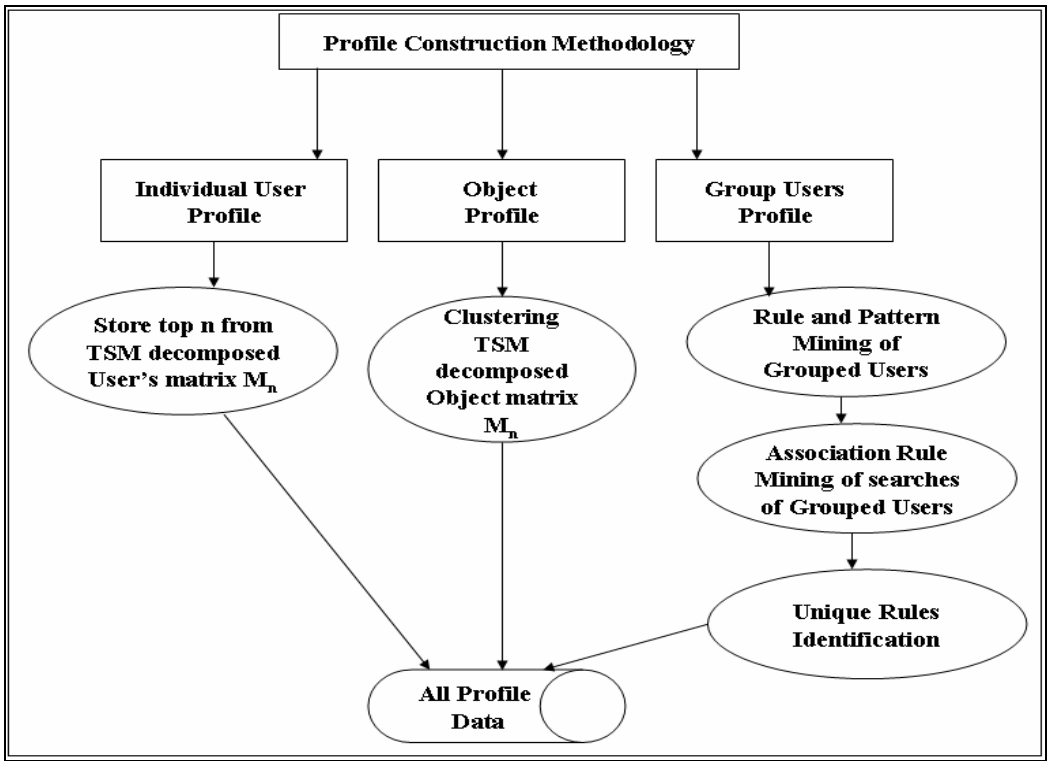

Figure 3.18 Complete User, Group and Object Profile Construction Model.

Once, clustering (Grouping similar users/objects) is done, the most important step for building group profiles and object profiles is finding patterns (associations) and unique rules between users and objects respectively. In the case of users it is done to find what objects interest a group of users, and in the case of objects, it is done to find similar objects based on their features, which can be recommended.

The various kinds of profiles discussed in this research are

1) Individual user profiles based on the two-dimensional methods.
2) Individual user profiles based on the TSM method.
3) Group user profiles based on the two-dimensional methods.
4) Group user profiles based on TSM method.
5) Object profiles based on the two-dimensional methods.
6) Object profiles based on the TSM methods.

3.3.1 Individual User Profile Creation Methods

In Web personalized systems, a user profile represents a ranked list of objects that interests a user, where such assumptions about user likings are deduced from his previous search information. User behaviour modelling based on multiple searched attributes is a complex problem. Various methods from vectors to matrices are currently used to find top rated features as searched by a user. However, due to the multi-dimensionality of Web log data, such information may be prone to loose latent relationships that exist between features when such data is modelled as a two-dimensional data. The two methods of creating user profiles, which are discussed in this thesis, are

1) Methods based on two-dimensional models and
2) Methods based on high-dimensional data models.

Creation of both of these methods is discussed below.

a) Individual user profile based on two-dimensional methods: An individual user profile represents a user's top interest, in one or many search dimensions. To create individual user profiles, the number of independent searches made by a user are utilized. Three methods using two-dimensional methods are proposed in this thesis for creating user profiles. These methods are:

1.) In the first method, matrix-based methods are employed to create user profiles. A matrix of identifed dimensions $M_1 \times M_2$ is created for each user, where M_1, M_2 are the selected dimensions. For example, in a car sales website, car make and model are basic features based on which cars are commonly classified and identified. The two-dimensional matrix is created for each user, representing the two searched dimensions as number of rows and columns. The term frequency denoted by $\sigma_{m_1 m_2}$ of such identified features as searched by a user is then input in respective index value $\mathbf{B} = [\sigma_{m_1 m_2}]_{M_1 \times M_2}$ of the matrix. Once a matrix of user searches is created, various matrix decomposotion techniques like SVD, PCA and NNMF are employed to find the top rated make and models. The highest *top* Υ values obtained after matrix decomposition are then saved in the user profiles.

2). In the second method all the features that are the part of a parameterized search query $(q_1, q_2 ... q_n)$ of a user are used to create search vectors of each user. Each vector

$iu_{z\eta}$ represents a user's interest. The idea behind the concept is to map multi-dimensional data into a two-dimensional space. The term frequency values using IDF (Inverse Document Frequency) for similar searches consisting of same searched parameters is ascertained. These vectors are then compared to find similar interests using cosine similarity. Dimension values from highly similar searches are then saved in the user profiles.

3.) In the third method to create individual user profiles, association rules from the total searches of a user are found out. Prominent features like make and model are taken for creating associations. For finding associations from user searches, only associations of length two are considered. Associations of length one are equivalent to frequency based evaluation methods. For associations greater then length two, the number of rules formed can be much less (since there may be users, who have made only a few searches as compared to other members of the group) also large number of associations are rare compared to association rules with small number of itemsets .

b) Individual user profile based on tensors: Individual user profiles built using tensors consist of all the highest rated interests of a user in each of the searched dimensions. Once the individual user model is created and decomposed (as explained in Section 3.1.2 and Figure 3.6), the *top* Υ values in each dimension are taken as the dimension values to be saved in the user profile. These values represent a user's top interest in each searched dimension. Unlike matrix or frequency based methods, tensors have the ability to maintain various relationships between different interacting dimensions (factors). This happens due to the the ability of TSM models to allowing free interaction between various components [1], thus minimizing information loss and maintaining inter-dimensional relationships. Tensor decomposition of a user's model also enables to highlight top rated objects as searched by a user highlighting the latent relationships that exixt between different dimension values.

Choosing the number of *top* Υ values to be saved for each dimension depends on the need of the website. This research work has taken each of the top Υ=3, 5, 10 and 15 values in each searched dimension to be saved in a user's individual profile. Detailed experiments and discussion on finding the optimized Υ value have been done in chapter 6 Section 2. The complete methodology of creating user profiles from tensors, once when the model is created and decomposed, is discussed in the following steps.

1. From the tensor decomposition of a user's tensor, retrieve the component matrices from $\mathbf{M}_1, \mathbf{M}_2$ to $\mathbf{M}_n$, where the highest scored values in each of the component matrix $\mathbf{M}_n$ represents a user's top interests in the respective dimension.
2. Thus, for each matrix from $\mathbf{M}_1, \mathbf{M}_2 .. \mathbf{M}_n$, arrange values in each row $m_{i \times r}$ in descending order. Retrieve *top* Υ values from each matrix $\mathbf{M}_n$.
3. Save *top* Υ values in the user profile Table with complete details of objects as retrived from the database.

As an example of creating individual user profiles for a user, this research work shows a sample user profile based on the PARAFAC decomposition, which has the highest searched dimension values for each dimension. The highest values represent a user's top interest in that dimension. In this example the three highest decomposed values in each dimension are taken for saving in the user profile. For the range of kilometres (KMS) dimension as shown in Table 3.6, it can be seen that highest PARAFAC decompostion value is 1.000 which corresponds to a user's preference for a car that has done between 10,0001 to 15,0000 kilometres. Similarly, from the decomposed values of car model dimension as shown in Table 3.7, it can be deduced that the specific user shows the highest interest in the car model '*300*' which is a Mercedes-Benz car, as this car ranks highest in the specified car model dimension.

Dimension=KMS		
Highest PARAFAC Values	**Corresponding Value**	**User Profile Ranking**
1.0000	kms>=100001 and kms <=150000	1
0.0213	kms >=50001 and kms <=100000	2
0.0117	kms >=25001 and kms <=50001	3

Table 3.6 A user's top-ranked interest for cars with KMS done.

Dimension=Car Models		
Highest PARAFAC Values	**Corresponding Values shown as (Make,Model)**	**UPF Ranking**
0.9806	Mercedes-Benz, 300	1
0.1961	Alfa Romeo, 147	2
0.0004	Toyota, Camry	3

Table 3.7. A user's top-ranked interest for car models.

3.3.2 Group Profile Creation Methods

Group profiles consist of the highest interests of a group of users, where the highest group interests are scored based on the popularity of the objects as searched within the overall group. Unlike individual user profiling, group profiling is a CF-based approach where similar users are clustered based on items they have searched or ranked. Group profiling is a popular approach for making cost effective recommendations. For making recommendations to users in a group, some of the popularly used methods are 1) recommending *top* Υ items as searched by users in a group; 2) finding associations of items as searched by a group; 3) finding top ranked items as ranked by many users in a group.

However, as discussed earlier, finding any similarity between users-items is a complex process. It involves analysing high-dimensional data with many searches made by users, and each search may be different from the other. Therefore, to construct group profiles for making efficient recommendations, this thesis has used tensor-based models coupled with extraction of unique rules from the associations in a group.

a) Group profiles based on vector and matrix methods: To create group user profile, the number of independent searches made by all users in the same group are utilized. All unique searches made by users in a group are identified. All searched parameters are identified. Frequency of all similar searches of each user is found. A similar search is a search where all queried components are the same even if a user may have searched the same components in different search sessions. The major difference between an individual user profile using vector-based methods and group profiles using vector-based methods is that, in a group user profile, each row represents a user and each different column represents a searched component as searched by the user. The matrix contains searches of all users, represented as row vectors. On the other hand for an individual user, the search matrix so formed, contains searches of a single user. Two methods are then used to create group profiles. 1). In the first method, matrix-based methods are employed to create group user profiles. Top rated searched objects are identified in each cluster using popular two-dimensional reduction techniques such as SVD, PCA and NNMF. Such highest *top* Υ values are then saved in the user profiles.

2.) In the second method, vectors of searches for all users are created. These vectors are then clustered to find similar users. The searches made by similar users are then used to find associations. Object features which can distinguish between two searches are then identified by these associations. For example in the car website prominent features like make and model are taken for creating associations, where associations of length two are only considered, as associations of length one are equivalent to frequency based evaluation methods and for associations greater then length two, the number of rules formed becomes less. The reason behind this is that there may be users who have made few searches and finding appropriate association rules in such searches is difficult and is not efficient.

Once associations between groups of users are ascertained, only rules with high confidence values are taken. From these rules, unique rules for each cluster of users are derived. For a cluster C_k, two rules $\text{Rule}_{r_1}^k = Q_i \Rightarrow Q_j$, and $\text{Rule}_{r_2}^k = Q_l \Rightarrow Q_t$, are considered as redundant when $(Q_i = Q_t) \wedge (Q_j = Q_l)$. If rules are redundant only one of the rules such as $\text{Rule}_{r_1}^k = Q_i \Rightarrow Q_j$, is considered. Finally these *top* Υ rules are saved in the group profile of each cluster with object identification, user identification and cluster identification details.

b) Group profiles based on Tensor Models

Once, after decomposing the group users tensor model, clustering is performed to find similar users based on their searches (as explained in Section 3.1.3). The next important task is to find interesting patterns within such a group of similar users. Within each group of users, associations between their searches are found out. These associations are further mined to derive unique rules and finally saved as *top* Υ interests of group users in their respective profile. To achieve this, the following three steps are performed.

1. Finding association rules between users in a group.
2. Finding unique rules from such users-users associations in a group.
3. Saving rules, cluster details, user details and objects details as a profile.

Each of these steps is discussed below in detail.

1. Identification of Associations in Groups of Clustered Users: All searches made by the users in a cluster are collected and the frequent associations based on desired query components (like make-model in case of a car website) are mined. The

components chosen for making such rules should be able to clearly identify object instances in the database. In our case, make and model of a car are taken, because these two search components can clearly identify objects. Thus, each cluster contains the searched parameters that are searched by the users of the respective cluster. Association rules are mined for each individual cluster. All searches made by group members, consisting of car make and model are passed as input for mining association rules. All association rules with high confidence values are considered. This research has considered associations of length two, as the occurrence of associations of length greater than 2 was very rare, especially when the number of users in a cluster was small. In cases where the length of association is 1, such rules were equivalent to the frequency based methods.

2. Unique Rules Identification: Once association rules are extracted from a cluster of similar users, unique rules are ascertained from these associations. Unique rules need to be extracted because association of item sets may be redundant. Similar searched item sets may be generated with different ordering of item sets by association rules, even though the searched item values are same. For a cluster C_k, two rules $\text{Rule}_{r_1}^k = Q_i \Rightarrow Q_j$, and $\text{Rule}_{r_2}^k = Q_l \Rightarrow Q_t$, are considered as redundant when $(Q_i = Q_t) \wedge (Q_j = Q_l)$. From the association rules of each cluster, the number of unique rules that can be retrieved depends on the need of the website or service provider. If the number of recommendations to be made is high, then large numbers of rules have to be generated. In all the experiments conducted association rules that can generate 3, 5, 10, and 15 unique rules for each cluster of users are taken. All unique rules with high confidence score are considered. These rules along with the scores are then saved in the group user profiles as *top* Υ rules for the respective cluster. Such *top* Υ rules can then be given as recommendations to each of the users belonging to a same cluster. The complete methodology for finding association and unique rules from a group of users is explained in the algorithm presented in Figure 3.19.

Let Υ be the number of association rules desired (top rules) representing clustered user's behaviour. Let n be number of rules mined. Let k be the cluster index, where a cluster is identified by C. Let $\text{Rule}_n^k : Q_i \Rightarrow Q_j$ be the rule mined for cluster C_k on item sets q_i and q_j such that $q_i \in Q_i, q_j \in Q_j$ and $Q_i \cap Q_j = \varnothing$.

Input: Searches consisting of query components q_n made by the users in a cluster.

Output: List of *top* Υ association rules for the cluster.

Begin

Step 1. // Find rules for each cluster C_k.

for each $C_{1...k}$

for $j=1$ *to* n // Extract *top* Υ association rules.

If confidence $(Q_i \Rightarrow Q_j) = 1$ then

$\text{Rule}_i^k : Q_i \Rightarrow Q_j;$

Else

$Max(confidence(Q_i \Rightarrow Q_j));$

$\text{Rule}_i^k : Q_i \Rightarrow Q_j;$

Top $\Upsilon : Q_{in} \Rightarrow Q_{jn};$

End for

k++;

End for

// Extract unique rules for each Cluster, two rules are not unique where

$\text{Rule}_i^k : Q_i \Rightarrow Q_l$ and $\text{Rule}_j^k = Q_j \Rightarrow Q_t$, when $i \neq j$,$l \neq$ t and $(Q_i = Q_t) \wedge (Q_j = Q_l)$;

Return: Cluster wise *Top* Υ unique rules.

End

Figure 3.19 Algorithm for finding associations from a group of users.

A sample of deriving top 3 unique rules from association rules is explained in the example below as shown in Figure 3.20.

Best rules found:

1. Model=X-Trail 17 ==> MAKE=Nissan 17 conf:(1)
2. MAKE=Nissan 17 ==> Model=X-Trail 17 conf:(1)
3. Model=Grand Vitara 16 ==> MAKE=Suzuki 16 conf:(0.89)
4. Model=Outback 14 ==> MAKE=Subaru 14 conf:(0.75)
5. Model=ML 12 ==> MAKE=MERCEDES-BENZ 12 conf:(0.65)

Figure 3.20. Sample of associations found out for a cluster.

For the association rules derived for a cluster of users, as shown in Figure 3.20, the top 3 cars recommended with make-model are: 1). Nissan-X-Trail , 2). Suzuki-Grand Vitara and 3). Subaru- Outback. All users with belonging to the cluster can be recommended these *top* cars.

3. Saving rules: All unique rules with high confidence score and containing different query search parameters are saved in an ordered manner. These rules can be recommended to all users belonging to the same cluster. Recommendations to group users can also be made based on their current search parameters, by utilizing a ranking function that matches similar objects in the group profile and scoring them higher.

3.3.3 Object Profile Creation Methods

An object profile is a collection of objects, which have many features or properties similar within the group. Object profiling is an efficient alternative for making recommendations to users who do not have profile information. Object profiling has been a popular approach, as it deduces items-items similarities based on features that can help in recommending similar items as searched by a user [183].

The other advantage of making object models is that unlike group user models, which cluster users based on certain features, the object model further assists in finding similar items based on other features, which are not considered when constructing the group users model. Use of clustering for finding similar users based on searches is the norm, but creation of such object models can further help to find the latent relationships that exist between different objects, but maybe were not specified by the users. As an example the group users clustering model initially groups users into a cluster based on only a few parameters like make, model, cost, search type of a car. Wheareas the object model further finds closer similarities based on the features of the car like kilometres, transmission, enginesize, fuel, location id. Thus the two models in synchronization will be able to give better recommendations to users.

Various popular approaches used for finding items-items similarity include vector and matrix-based methods [128] employing Euclidian, Manhattan and cosine similarity measures. However, tensor-based methods have seldom been used to find such item-item similarities. The next section discusses about creating object profiles based on the popular approaches as well it discusses about the novel method of creating object profiles based on the tensor models

a) Object Profile Creation based on Vector and Matrix Method: To create object profiles various vectors consisting of prominent features of objects as identified are formed. Let each vector be represented as $a_1, a_2..a_n, o_z$ where $a_1, a_2..a_n$ are the object features and o_z is the object identifier which distinguishes each object instance from the other. Once vectors of all instances are created, clustering is done to group similar objects.

Step 1: Identify the number of clusters C_k to form, when grouping similar objects.

Step 2: For each cluster from $C_1...C_k$ retrieve the feature values, with cluster identification and object details. Save object clusters with cluster identification

details, object instance details and features of each instance, based on which clustering is achieved.

Once all information is saved, it can then be used for recommending similar objects as searched by a user. Users searches are compared with clusters and only similar objects in a cluster that match highly to a user's search are recommended.

b) Object profile creation based on the tensors method: For creating object profiles from tensor models, the values of decomposed matrix $\mathbf{M}_n$, (as explained in Section 3.1.4 Figure 3.10) are taken. All objects with near similar features are then grouped into respective clusters by clustering the values of matrix $\mathbf{M}_n$. Similar objects clustered in a group can be recommended to users, whose searches match objects in a cluster.

Step 1) Identify the number of clusters to form, and for each cluster from $C_1 ... C_k$ retrieve all features, with cluster identification details.

Step 2) Find patterns within clusters of objects and rate frequent item sets higher. This thesis has adopted CF approach, according to which top rated (searched) objects in a group obtain a higher score in each dimension. To find the top-rated items in each searched dimension, the number of times a dimension value was searched is taken into consideration. Dimension values with higher frequency are rated higher than other values.

Step 3) Save object clusters with cluster identification details, object instance details and top rated feature values for each cluster. Once all information is saved, it can be used for making recommendations. Table 3.8 represents a sample format in which object profile information, representing the three top-rated features for each object cluster can be saved. As an example shown in Table 3.9, the cars which have been clustered in a cluster C_k are shown and how values for cluster C_k would be represented in an object profile (Table 3.10). In Tables 3.8 & 3.10 *Dimensionid* is the dimension value corresponding to which the top 1-3 values are determined.

Field	Data type	Description
ClusterID	varchar(50)	Object cluster Id.
ObjectsId	varchar(5000)	All objects clustered in the group
DimensionId	int	Feature or searched parameter Id.
Top1	int	Feature Id of Top 1 value.
Top1Score	Float	Score of Top 1.
CorrespondingValue1	varchar(150)	Top 1 Feature value.
Top2	int	Feature Id of Top 2 value.
Top2 Score	Float	Score of Top 2.
CorrespondingValue2	varchar(150)	Top 2 Feature value.
Top3	int	Feature Id of Top 3 value.
Top3 Score	Float	Score of Top 3.
CorrespondingValue3	varchar(150)	Top 3 Feature value.
ProfileUpDate	Date/Time	Profile recent updated date.

Table 3.8 A typical object profile structure.

CID	Make	Family	Body	SYear	Cost	Kms	TM	Eng	Fuel	Loc
C_k	Toyota	4 Runner	4D Wagon	1995	10990	138000	5M	3.0	UNL	3
C_k	Toyota	Camry	4D Sedan	2004	14990	134125	4A	2.4	UNL	9
C_k	Nissan	Pathfinder	4D Wagon	2003	23990	63867	4A	3.3	UNL	3
C_k	Holden	Commodore	Wagon	2002	12990	89572	4A	3.8	UNL	5

Table 3.9 Example of an object cluster C_k, where CID-=Cluster id.

Sno.	Field	Dimension 1 (Make)	Dimension 2 (Model)	Dimension 3 (Bodytype)	Dimension 4 (Cost-Ranges)
1	**ClusterID**	2	2	2	2
2	**ObjectsId**	T1, T2, N1, H4	T1, T2, N1, H4	T1, T2, N1, H4	T1, T2, N1, H4
3	**DimensionId**	D1	D2	D3	D4
4	**Top1Id**	T1	T1M1	B1	C3
5	**Top1Score**	1	1	1	0.75
6	**CorrespondingValue1**	Toyoya	4 Runner	4D Wagon	10-15K
7	**Top2Id**	N1	T1M2	B2	C4
8	**Top2 Score**	0.75	0.84	0.87	0.65
9	**CorrespondingValue2**	Nissan	Camry	Wagon	20-25K
10	**Top3Id**	H4	N1M1	B3	C5
11	**Top3 Score**	0.50	0.49	0.74	0.56
12	**CorrespondingValue3**	Holden	Pathfinder	4D Sedan	30-35K
13	**ProfileUpDate**	01/01/2010	01/01/2010	01/01/2010	01/01/2010

Table 3.10 Example of an object profile for cluster C_k, showing top three values for four dimensions.

3.3.4 Discussion: User Profiling Methods

This section discusses about the various profiling methods based on vector, matrix and tensor methods. User and object profiling are the core part of any personalization method. Efficiency and effectiveness of recommendations depends on the quality of user profiles created. Better user or object profiles created using items-items or user-items similarity methods result in good quality of profiles.

Overall, once when all the user profiles are created, they can be saved in any desired format. A sample of such user profile format is shown in Table 3.11 below. Here the top three items based on user profile information can be.

Field	Datatype	Description
ClusterID	varchar(50)	Cluster Id in case group profile is used.
IP	int	Unique user identification number.
Identifier	int	Status, whether value from UPF or GPF.
DimensionId	int	Feature or searched parameter Id.
ThresholdStatus	Boolean	Limit for no. of recomm. been set or not.
RecThreshold	int	Upper Limit for maximum recommendations in this dimension.
Top1	int	Feature Id of Top 1 value.
Top1Value	Float	Score of Top 1 Interest of user.
CorrespondingValue1	varchar(150)	Top 1 Feature value.
Top2	int	Feature Id of Top 2 value.
Top2Value	Float	Score of Top 2 Interest of user.
CorrespondingValue2	varchar(150)	Top 2 Feature value.
Top3	int	Feature Id of Top 3 value.
Top3Value	Float	Score of Top 3 Interest of user.
CorrespondingValue3	varchar(150)	Top 3Feature value.
ProfileUpDate	Date/Time	Profile recent updated date.

Table 3.11 A typical user profile structure.

	Field	Dimension 1 (Make)	Dimension 2 (Make)	Dimension 3 (Model)	Dimension 4 (Model)
1	ClusterID	C_2	C_2	C_2	C_2
2	IP	IP1	IP1	IP1	IP1
3	*Identifier**	*1*	*2*	*1*	*2*
4	DimensionId	D1	D1	D2	D2
5	RecThreshold	5	10	3	4
6	Top1	T1	H1	T1M1	H1M1
7	Top1Value	1	0.88	1	0.85
8	CorrespondingValue1	Toyoya	Holden	Camry	Astra
9	Top2	N1	F1	N1M1	F1M1
10	Top2Value	0.75	0.71	0.84	0.65
11	CorrespondingValue2	Nissan	Ford	Patrol	Falcon
12	Top3	H1	HO1	H1M2	HO1M1
13	Top3Value	0.50	0.18	0.49	0.34
14	CorrespondingValue3	Holden	Honda	Commodore	CRV
15	ProfileUpDate	01/05/2010	02/05/2010	01/05/2010	02/05/2010

Table 3.12 Example of an User profile for user u_z, showing top three values for four dimensions.

The Table 3.12 represents top three interests of a user u_z, based on his individual profile (Identifier*=1 in Table 3.12) and group profile (Identifier*=2).

Once user profiles are created, they can be shared across various websites or agents as ontology. As an example as shown in Figure 3.21, a user profile created in RDF for a test website can be used by news or media websites, to recommend news about sports-cricket to the user. Once all categories are available as ontology in RDF format,

it would mean that the user profile has to have categories of interests in RDF format that can be matched with a user's interest category. To achieve this a Semantic Parser Engine (SPE) can be created. The role of SPS would be to classify a topic or a website page under a category. Each time the string or query is passed to this SPE, it will classify that topic into its subsequent category automatically.

Eventually, the user profile knowledge can be used by various Web agents to make recommendations to different users, especially in Web 2.0 where such knowledge can be shared across groups of users such as friends in social networks. In Web 3.0 such ontologies can interact with various information agents to extract the relevant information as may be desired by a user.

```
Sample of User Profile in RDF:
<? Xml version="1.0"?>
<rdf:RDF
Xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
Xmlns:UserProfile=http://www.abc.com.au/UserProfile#>
<rdf:Description
rdf:about="http://www.abc.come.au/UserProfile/IP_ADDRESS1">
< UserProfile:Country > Australia < /UserProfile:Country>
< UserProfile:Location > Brisbane < /UserProfile:Location>
< UserProfile:Age > 20 < /UserProfile:Age>
< UserProfile:Sex > Male < /UserProfile:Sex>
< UserProfile:Sports >Cricket < /UserProfile:Sports>
< UserProfile:ECommerce > Electronics < /UserProfile: ECommerce >
</rdf:Description
```

Figure 3.21 A sample user profile in RDF for a demo website.

Thus, in general even though user-profiling needs extra computation resources like processing time, space and memory to process for creating and saving a large number of user profiles. These overheads can be compensated by creating the user profiles offline and utilizing these profiles dynamically in real time. The other advantage of creating these profiles is that rather than giving plain search results to users, these profiles can be used to give personalized search results to users utilizing their profile information. If there are z number of users, then the complexity of creating individual user profile is $T(z) = O(z)$, however, in case of group user profiles the complexity reduces to $T(C_k) = O(z'), where\ z' << z$. Here z' is the number of users in the cluster C_k.

3.4 Recommendations based on user, group and object profiles

Objects can have multiple features, based on which a user's searches are made. Traditional CF methods employed for finding similarity between users-items ignore the multi-dimensional nature of search log data and are unable to recommend unique objects to different users [128]. These methods consider an item as an object, whereas, the item may be a combination of many features, represented as a vector itself. Finding the latent relationships between a user's searches and an object's features is often ignored. These relationships could be answers to simple questions like which two users are similar or more complex like which two searches are related and what searched features are more similar.

In this thesis a novel recommendation method for effective Web personalization, which uses the implicit information about users by using the search log data is proposed. The methodology utilizes the search log data to infer the user rating about objects. This is a collaborative approach, which group's users according to their common searches and then finds users–items correlations within a group. To find the correlations between users according to their usage of objects, a tensor based high-dimensional data model is employed. Three profile building strategies like the individual, the group and the object profiles using tensors are proposed. To our best knowledge a recommendation model using various profiles built using tensors from users search log data, consisting of the multiple search parameters and users as different dimensions, and having greater than three dimensions, has never been proposed.

3.4.1 Recommendation based on individual user profile.

The major objective of creating individual user profiles is to help a user find the relevant information that he may be looking for. This information can be in the form of recommendations. To make recommendations to an individual user, his user profile information is used. The profile has information about the highest interests of a user in each searched dimension. All interests of a user are placed in a descending order of relevancy. Thus, result sets matching his *top* Υ profile interests can be found out and similar objects can be given as recommendations to a user. In case of an individual user, a better recommendation model limiting such recommendations to *top* Υ can be

found out. If the quality of recommendation starts declining, a cut-off limit for the number of recommendations to be made to a specific user has to be ascertained. Such cut-off limits can be set based on the datasets and evaluations of previous recommendations made to the user. The proposed recommendation algorithm for an individual user is discussed in Figure 3.22. The algorithm uses a $Retrieve(u_j^{\Upsilon})$ function that helps in retrieving the *top* Υ results from the user profile and gives such results as recommendations to the user. The function uses a SQL $top(\Upsilon)$ function to retrieve top results from the database where the user profile is stored. The function takes Υ the desired number of top rated values in each dimension and from the user profile finds all top rated values in each dimension from $M_1, M_2..M_n$. The other two important variables are #*TN* (Threshold Number) and *TS* (Threshold Status). The Threshold Number is the cut-off limit for the number of recommendations to be made to a user in a dimension, while the Threshold Status (*TS*) acts as flag to signify whether such threshold limit has been set up for the particular user or not. The other function that the algorithm performs is a $Display\ (u_j^{\Upsilon})$, function that displays *top* Υ recommendations to a user u_j .

Input: User identification u_j, Υ number of top recommendations desired.

Output: List of *Top* Υ Recommendations.

Let *#TN* is threshold number and *TS* is threshold status.

Begin

Step 1. *Retrieve* (u_j^{Υ}) ; //Get highest scored values in each dimension $q_1, q_2 .. q_n$ from user profile.

TS:=True;

#TN=1;

Step2. //Check Threshold.

For $\Upsilon = 1..k, i = 1..n$

If *TS=True* then

Do while $k \le \#TN$

Retrieve u_j^i {

top (q_i^{Υ});// Retrieve desired number of highest scored dimension values.

}

$\#TN++$;

End do;

Else

Retrieve u_j^k {

top q_i^k ;// Retrieve highest scored dimension values.

}

End if

End For;

Step 3.

Display u_j^{Υ} {

select $q_i, : i = \Upsilon$; //*Displays all top retrieved results.*

}

End

Figure 3.22 Recommendation algorithm for a user based on his profile.

Take for example a user profile that has the following information (Table 3.13). The user has searched used cars from the cars database, and based on his searches a profile is created, which clearly outlines important searched dimensions. From his user profile, it can be clearly deduced that the user is looking for a used car, with a body type of SUV and a cost range of $19000-25000. Furthermore, the user would highly prefer a Holden, Toyota, Honda, Ford and Nissan in that order. Finally, for matching results, based on his profile information and current information needs, a ranking method is needed that can utilise the profile information and rank relevant items based more highly on the user's current needs.

Top *n*	Car Makes	Score Make	Models	Score Model	Body type	Score Bodytype	Cost ($)	Score Cost
1	Holden	1	Captiva	0.75	SUV	1	25,000	1
2	Toyota	0.9	Rav 4	0.71	SUV	1	25,000	1
3	Honda	0.81	CRV	0.70	SUV	1	19,000	0.87
4	Ford	0.75	Falcon	0.56	Sedan	0.78	20,000	0.95
5	Nissan	0.71	Murano	0.45	SUV	0.45	25,000	1

Table 3.13 A snapshot of some dimensions from the *top* Υ objects saved in a user profile.

3.4.2 Recommendation Based on Group Profiles.

Individual user profiles have the ability to give more personalized recommendations as each individual's preferences are taken into consideration. However, a major drawback when making recommendations to a user based on his individual profile is that a user may not be recommended unique items, which are similar to his search. This can happen because a user may not have searched such items on his own. Therefore, to make the recommender more effective, group profiles can be used to recommend similar items. The other advantage of using group profiles for making recommendations is that creating individual user profiles is costly.

Recommending items based on group profiles is a cost-effective way of improving recommendations. Once the group profiles have been created, any associations between the searches of users and the same clusters can be determined. Unique rules, based on that association, consisting of searched objects can be saved in the group profiles of users.

The next step is making recommendations to users in a cluster. All *top* Υ rules that have the high confidence values are taken. Here Υ is chosen so that the number of unique rules that can be derived from associations is at least equal to fifteen rules per cluster. Only the top fifteen rules are chosen for recommendations because a higher number of recommendations decreases the quality of recommendations [76]. The other drawback of giving too many recommendations is that a user looses interest in recommendations if large numbers of objects are given as recommendations. The methodology behind recommendations based on a group profile is similar to recommendations based on individual profiles, except that in the case of the group profiles, recommendations are made based on common search interests of users in a

group, unlike individual profile method, where recommendations are made based on the individual interests of a user. The other difference is that unlike individual profiles, group profiles also cater to information needs of an anonymous user. For users with no profile, rating or search data, it becomes difficult to recommend objects. To avoid this *cold-start* problem, in case of an unknown user, his searches are matched with a group profile and similar searches from a group which matches his interests are recommended. The algorithm uses a $Retrieve(u_j^{\Upsilon})$ function that helps in retrieving the *top* Υ results from the group user profile and gives such results as recommendations to the users. The other function that the algorithm performs is a $Display\ (u_j^{\Upsilon})$, function that displays *top* Υ recommendations to users. The whole recommender process is explained in the algorithm in Figure 3.23.

Let the searched dimension values be represented as $q_1, q_2..q_n$.

Input: User identification u_j and whether known (KU) or anonymous user (AU), and *n* the number of recommendations desired.

Output: List of *Top* Υ Recommendations R_n.

Let *#TN* is threshold number of recommendation for a user u_n and *TS* is threshold status.

Begin

Step 1. Identify User u_j, whether KU or AU.

Step 2. //Recommending to a known user.

If u_j=*KU*

Step 3. *Retrieve* (u_j^{Υ}) ; //Get highest scored values in each dimension $q_1, q_2..q_n$ from user profile.

TS:=True;

#TN=1;

Step4. //Check Threshold.

For $\Upsilon = 1..k, i = 1..n$

If *TS=True* then

Do while $k \le \#TN$

Retrieve u_j^i {

top (q_i^{Υ});// Retrieve desired number of highest scored dimension values.

}

$\#TN++$;

End do;

End if;

Else

Retrieve u_j^k {

top q_i^k ;// Retrieve highest scored dimension values.

}

End For;

End if;

Step 5.

Display u_j^{Υ} {

select $q_i, : i = \Upsilon$; //*Displays all top retrieved results.*

}

End

Figure 3.23 Complete recommendation algorithm for a group of users.

3.4.3 Recommendation based on object profiles

Object profiles store similar items that have very similar features or are rated similarly by many users. Recommendations based on ratings of objects are common methodology [99], [150] but it creates problems when new objects without ratings information are introduced. The new object problem can be resolved by alternatively creating object profiles based on features of objects. One of the main objectives of creating object profiles is to cater to the information needs of an anonymous user. An anonymous user is a new visitor to the website, who does not have profile information.

Recommending similar items as searched by a user becomes effective when good object profiles with high similarity between objects are created. When an anonymous user visits a website and directs a specific query to the search engine of the website, the query parameters can be matched with object properties in the stored object profiles. Based on the number of features that match the user's query, object clusters with highest score can be created and objects from the respective cluster can be recommended to the user.

The other advantage of creating object profiles while making recommendations is that, when a user with profile information makes a search, he can be recommended unique objects that match his search but which differ from his previous searches. This aspect of recommendation can bring greater interest and variety to the recommendations made to a user. The user no longer has a limited vision about searched objects, but can be recommended new objects that are similar to his searches.

To explain how recommendation can be made based on object clusters, recommendation is explained with an example. For each cluster in C_k, identify the top scored values in each dimension from $M_1, M_2..M_n$. Retrieve cluster details where dimension values for a user's search query $(q_1, q_2..q_n)$ have highest values. Once an object cluster, consisting of cars is saved as shown in (Table 3.14), and if a user's search for parameters *(Make / Model / Cost Range)* is say *(Toyota / RAV4 / 1-5000),* then the results that can be returned to such a user can be in the following format - *(Cluster I.D., Car I.D., Score)* as *(C_1,1,0.93), (C_1,2,0.33), (C_1,3,0.28)*. Here score can be evaluated based on a number of features matched with a user's search. Hence, similar cars clustered on the basis of make, model, and body type and which meet the search criteria of a user, are displayed to a user.

Cluster Id	**Car id**	**Make**	**Score Make**	**Model**	**Score Model**	**Cost $**	**Score Cost**	**Avg. Score**
C_1	*1*	*Toyota*	*1*	*RAV4*	*1*	*4000*	*0.80*	*0.93*
C_1	*2*	*Honda*	*0*	*CRV*	*0*	*5000*	*1*	*0.33*
C_1	*3*	*Ford*	*0*	*Territory*	*0*	*6000*	*0.83*	*0.28*
C_2	4	Honda	0	City	0	9000	0.56	0.19
C_2	5	Holden	0	Comm.	0	20000	0.25	0.08

Table 3.14 Sample Table showing how Objects clusters are saved.

Thus, the cars that can be recommended to the specific user are cars in cluster C_1 and the *top* Υ objects in the respective cluster can be used for making recommendations.

3.4.4 Summary: Recommendation Methodology

This section of the thesis discussed about the methods of making recommendations to individual users (based on their profile information), group users (based on their groups and unique rules extracted from the subsequent associations of user's searches in a group) and object profiles (based on items-items similarity, evaluated after finding relationships between the various features of object). Recommendations based on an individual user's profile are costly as each visitor's profile has to be created and saved. A large number of individual profiles may raise computational complexities in terms of space, memory and time.

The type of recommendation methodology adopted may depend on the needs of a website. In the case where there are a large number of unregistered users, then group user profiles or object profiles can be utilized by the website. However, when there are no such issues, individual user profile can be used for making quality recommendations. If all three methods of recommendations are used in conjunction then the quality of recommendation could be improved.

No matter what methods are used for making recommendations, one crucial factor in deciding the quality of recommendations is choosing the number of recommendations to be made to a user. Recommending a small number of highly related objects is better than recommending a large number of unrelated objects. Keeping this in consideration all the methodologies proposed for making recommendations adopt tensor methods for finding highly co-related users and objects.

Once the number of recommendations to be made to a user is decided, the other issue that can improve the performance of a recommender system is choosing how these *top* Υ results should be presented to a user and how the user's current search needs will be incorporated along with his past preferences by the personalized system. To achieve these objectives a ranking methodology is proposed in the next section.

3.5 Ranking Methodology for Web Search and Recommendations

Searching is one of the prominent and most widely utilized activities for finding the relevant information from the Web databases. A user query can be based on a single feature or a group of available features in the database. At times when the user query is specific, very few or no results are returned, as not all query conditions are matched. On the other hand, when a generic query is asked, too many results (or rows) are returned. These results are not ranked in the order of relevancy to the user. In these circumstances, it becomes difficult to utilize relevant information from the Web databases. Moreover, the standard database search does not allow identifying the highlighted features that are searched by the user and utilize them to rank results based on importance of these features. The problem of optimal search becomes more critical for the Web-based databases. Most of the Web data resides on large databases and requires only relevant results to be returned. The proposed method focuses on ranking the results retrieved for a Web database information system.

This section proposes a ranking algorithm, called the Features Importance Technique (FIT), utilizing the importance of searched features based on a users' past behaviour. Experimental results with 'Mileage *Cars*' server log data confirm the effectiveness of ranked results when this method is used to rank users queries. Results show that this method can effectively handle the problem of too many results returned. In case of specific queries having no results returned, the proposed method can do ranking based on the number of features matched. This method also extends the use of this algorithm to returning personalized search results to visitors of a Website.

3.5.1 Need for Ranking Recommendation Results

Recommending algorithms helps to identify a user's *top* ϒ interests based on his past search behaviour. These interests are saved as user profiles. However, when displaying results to a user current search criteria along with his profile information have to be kept in consideration. A ranking function is necessary to present the user with an ordered list of recommended results. The major objective of a ranking function is to present highly relevant items higher in the recommendation list. When making recommendations or presenting results to a user based on his current search

and profile information, all results which have to be presented and ranked have to be queried from a data warehouse.

Since, all information resides in various databases, a common problem faced in information seeking from databases is that either too few or too many results are returned. In both cases query reformulation is the normal behaviour of Web users [83], which clearly indicates that the returned results do not satisfy the user's information needs. From the data perspective, the possible scenarios are either that there are no relevant records to be found, or there are too many records that match fully or partially a user's searched query.

Let W_{db} be a Web database having a parameterized search Q that contains a maximum of $\{q_1 .. q_n\}$ features. A feature q_n can be of data type numerical, binary, and categorical or text. A query Q, over W_{db} can be represented with a conjunctive selection condition of the form $Q = \wedge_{i \in \{1,..k\}}(Q_i \ \theta \ q_i)$ where $k \leq n$ and θ are various query selection criteria's represented as $\theta \in \{>, <, =, \neq, \geq, \leq\}$. The values of these features from $\{q_1 .. q_n\}$ may further be selected based on various operators like $\{and, or, not, like, in, between\}$. A fully matched result set means if a user has searched for $\{q_1 .. q_j\}$ parameters, where $n = 1..j,..n$ and $\forall\, j \leq n$ then the returned result sets matches all $\{q_1 .. q_j\}$ searched parameters. A partially matched result sets occur when only few query parameters $(\bar{j})$ of a user's query match the returned results sets. Let us consider an example. If Q_1 is a query with q_n searched parameters and is represented as shown in Equation (27), then

$$Q_1 = \theta(q_1 \cap q_2 ... q_n). \tag{27}$$

Due to limitations of search results being displayed by such queries they have been called as strict queries and the output of such queries will be any of the two cases as shown in Equation (28).

$$\text{Output}_{Q_1} = \begin{Bmatrix} 0 = \text{No records returned} \\ j = \text{Number of Fully matched records} \end{Bmatrix}. \tag{28}$$

When a user directs such a query to a query engine, due to strict query conditions very often no result sets are returned. However, there may be other attributes present in the database, which had matched few but not all the attributes of the user's query. In such

a scenario, finding the right information becomes a cumbersome process. Users have to formulate the same query many times and search these unranked results to obtain the most relevant information that he or she needs.

Now consider queries with the following conditions (Equation (29)).

$$Like\left(q_1 \; or \; q_2 \; or \; q_3 \; or .. \; q_n\right). \tag{29}$$

In this thesis such queries have been called as flexible queries (relaxed queries) and the output of such queries will be any of the three conditions (Equation (30)):

$$\text{Output}_{Q_2} = \left\{ \begin{array}{l} 0 = \text{No Records} \\ j = \text{Fully Matched Records} \\ \overline{j} = \text{Partially Matched Records} \end{array} \right\}. \tag{30}$$

In contrast to strict queries, the flexible queries return a large number of result sets, resulting in the problem of too *many-answers*. It would be too cumbersome to find the most relevant and highly matched results as needed by the user. Various methods like query reformulation [139] and query relaxation [112], [124] have been proposed however these methods have not eased the burden of searching the most relevant results as needed by a user. These methods may partially resolve such problems but considering the nature of search queries given by users a method that fits in all circumstances and is flexible is still missing. Query reformulation may often distract the user from the actual search needs while query relaxation may again return too *many-answers*.

The proposed Feature Importance Algorithm (FIT) promises to handle these problems effectively. It does rankings of results based on the query, and user needs. In both cases when there is a necessity to implicitly identify relevant features from workload or to explicitly define preferences for certain features, the FIT algorithm is equally effective. Unlike most of these previously discussed methods, which either use query relaxation or require large workload information, FIT offers a lot of flexibility in terms of its applicability and implementation. In cases where no workload or user profile information is available explicit categorization provided by a user or collaborative information (i.e. depicting important searched features) can be used to rank most relevant results as needed by a user. In the worst case when no information is available, FIT retrieves and scores tuples based on the number of features matched to a user's query.

3.5.2 The Proposed Ranking Methodology for Returning Personalized Search Results

When making a search, a user may select any number of features from the available list of features or parameters. Consider the ideal ranking plot of searches. If *FS* is the Number of Features searched by a user and *FM* is the Number of Features matched to a user's query then top ranked result sets are returned when FM=FS which can be clearly seen from the Figure 3.24 of rankings plot.

$$\text{Ideally Top Ranked results are when } \frac{FM}{FS} = 1 \tag{31}$$

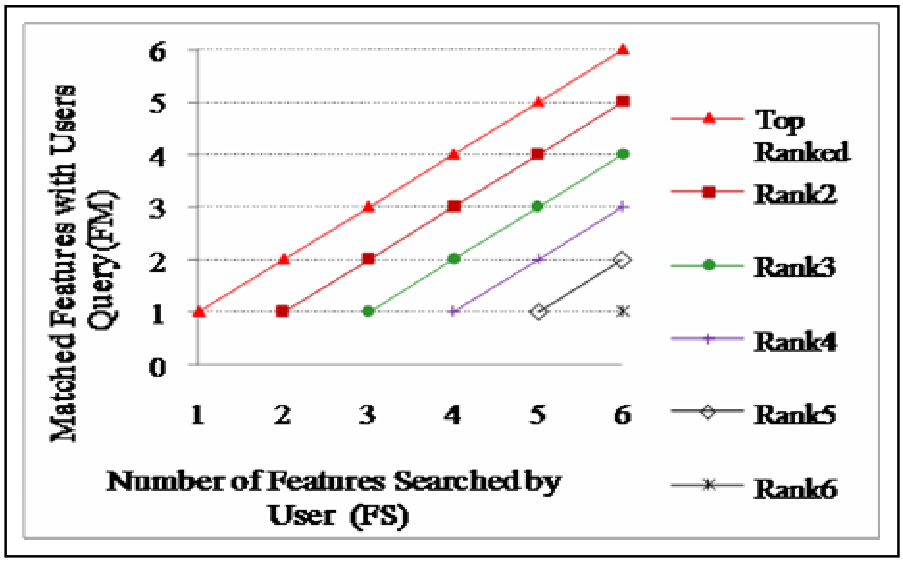

Figure 3.24 Ideal Ranking plot.

All results that match exactly the features searched by a user are ranked more highly. When ranking these results, individual search preferences could be deduced on the basis of workload, explicitly specified by a user or could be deduced on the basis of a user's profile. In the case where $FM < FS$, there can be two cases.

The first case is when each attribute or feature present in the user's search query is given equal importance. In this case, the order of the matched feature does not play any role in ranking, but the number of matched features plays a vital role in the ranking of result sets. All result sets which have the same number of matched features, will be ranked equally, irrespective of what features are matched with the user's query. This kind of ranking can be ideal for database ranking where knowing the importance of features for a particular user is not easy (of course unless specified

explicitly). Ideally, in this case all searched parameters requested by a user can be given equal importance.

The second case is when the searched features of a user's search are given preferences. This preference of features can be deduced by analysis of user's profile or by collaborative filtering (from Web log data). This is similar to a conditional search where a user specifies to which parameters or features to give higher importance when displaying result sets.

Consider the $(q_1 ... q_n)$ search features or parameters in a Website. In a database, these features may be columns in one Table or multiple Tables. For example, for a car sales website like '*Mileage Cars*' the features may be car make, car model, car body type, car cost and so forth. If n is the number of total columns (features) then consider v a variable which depicts the proportionate representation of each feature and whose value depends on the number of total features. It is calculated as shown in Equation (32).

$$v = \frac{1}{\text{Total No of Features (n)}}. \tag{32}$$

The value v is used to get a proportionate distribution of each feature in the overall search query. It is ideally representing the individual impact of each feature. Let α be a preference variable, which can be same for all features or can be set differently for all features. In case when multiple parameters are searched, the weight of important features as searched by a user can dominate the value of $v,$ and results can be too biased, hence to normalize these values α is used. The value of the preference variable α can be set manually or can be set empirically depending upon the importance of each feature. Usually α should be set a fraction higher than the value of v. Ideally, the value of the preference variable can be represented as shown in Equation (33).

$$\text{Preference Variable } \alpha = (v + \frac{v}{2}). \tag{33}$$

The next important variable which can be set is the Importance Number of a feature and is denoted by I_n. This number shows the important role a feature plays when a user searches. I_n indicates the priority of a feature while making a search. For certain features, it can be set as the same, as they play an equal role in the overall search. This

value can be set by identifying the search behaviour of a user from the Web log data (Equation (34)). The more the feature is used in past searches, the higher the value it is given.

$$\text{Searched Ratio}(C_n) = \frac{\text{Total number of users used the feature in searches}}{\text{Total number of searches in the database}} \quad (34)$$

Features are ranked according to the descending order of their searched ratio. The highest searched feature gets an importance value (I_n) of 1. A value of 2 means it is the second most important feature and so on. Consider an example. This research work has used the Web log data of a popular car sale website in Australia. The server log data analysis shows that the most important features for the users were the make, model and cost, and around 87% of users had at least searched for these three features from the available search options. Table 3.15 shows the search analysis.

Imp. No. (I_n)	**Feature's Name**	**Data Type**	**% Searched by all users (WebLog Data)**
1	Make	String	99.9%
2	Model	String	99.8%
3	Cost	Numeric	60.5%
4	Body type	String	10%
5	Engine size	Numeric	7.1%
6	Fuel	Binary	1.78%
6	Safety	Binary	1.65%
6	Comfort	Binary	1.57%

Table 3.15 An example of Importance Number of Features for a car sales Website.

The most important search query feature is 'Make' (thus here in this case study, it has been assigned an Importance number of 1) and so on. For the three search features Safety, Fuel and Comfort, the I_n value can be set as the same or can be set as per a user's preference. In this example it has been set as same as they carry little significance in comparison to other parameters.

Let μ_n be the combined ranked score of one search result set, consisting of n number of searched features as searched by a user. For a textual and binary search feature (E.g. make, model, safety rating of the car), the score is represented as ${}_t\mu_n$ and is evaluated in cases where in the number of features searched $n \geq 2$ (Figure 3.25).

$$ {}_t\mu_r = \left(\text{No. of Total Features Searched} - \text{Importance No. of Searched Feature}\right) \times \left(\text{Preference Variable} - \frac{1}{\text{Number of Features}}\right) $$

Or

$$ \sum {}_t\mu_r = \{(n - I_r) \times (\alpha - v)\} $$

Figure 3.25 Ranking method of textual and Boolean features.

However, when the number of features searched is 1, then equation in Figure 3.25 reduces to $\sum {}_t\mu_n = (\alpha - v)$. The equation in Figure 3.25 is a form of normalization which shows that if a user searches for a categorical feature that matches exactly the available value in the database, then based on the importance of the feature to the user (represented by the importance number (I_r)), and relative to the number of features a user has searched (n), what is the normalized score of overall feature, that can distinctively identify the attribute and its impact in the overall search of a user.

For the numeric features (E.g. cost, engine size of a car) the proportionate similarity between two compared numeric attributes is calculated. For such features the ranked score denoted as ${}_b\mu_n$ is calculated based on the value of the searched feature and its proportionate similarity with other numeric attributes in that domain. If ${}_u q_l$ is the numeric attribute searched by the user and ${}_d q_l$ is the corresponding database value that has to be compared with this, then it is evaluated as

$$ {}_b\mu_n = if\ {}_d q_l > {}_u q_l \text{ then} \left(\frac{{}_u q_l}{{}_d ql} \times v\right) \quad \text{else} \quad {}_b\mu_n = if\ {}_u q_l > {}_d q_l \text{ then} \left(\frac{{}_d q_l}{{}_u q_l} \times v\right) $$

Figure 3.26 Ranking method of numeric feature.

Thus, the equation in Figure 3.26 compares the two numeric features relative to all the other features searched. In cases where two compared values are the same the overall score of the tuples that exactly match these searched values would increase drastically. In such a scenario regardless of importance number of all categorical features, the ranking would be biased towards such numeric features. Hence, to remove this biasness and to normalize the attribute score, such attributes are multiplied by the proportionate representation of each feature (v).

Thus the scoring that is obtained after this is approximately equivalent to the scoring that is obtained for a single categorical attribute. In such a scenario, the only factor that can dominate the search is the importance number of the searched feature or attribute. The complete methodology of Feature Importance Technique (FIT) based Ranking Algorithm [143] is discussed in Figure 3.27. Once the numeric and categorical scores of database attributes matching a user's query are evaluated then these scores are added. Each summarized score represents a score of one single database tuple that has some similarity with the user's query. Once, all scores are calculated, normalization is executed as shown in Equation (35).

$$\mu_k = \frac{\mu_k}{\mu_{Max}} \tag{35}$$

where μ_{Max} is the highest score of all retrieved rows and μ_k is the score of k^{th} row.

Input: Query with parameters/features as searched by a user.
Output: Ranked result sets matching users query and interests.

Let m be the number of total features and n be number of features as searched by a user. Let k be the number of tuples to be compared with users search and let I_n is the importance number of each searched feature.

If ${}_uC_n$ be users search query and ${}_dC_n$ be result sets to be compared with users search query.

Begin

Step 1. Calculate variable $v = 1/m$.

Step 2. Evaluate Preference Variable $\alpha = (v + \frac{v}{2})$

Step 3. // Compare all features as searched by user.

for i=1 to k

for j=1 to n

// Retrieve I_r of each Search feature ${}_uC_i$.

// For text or binary feature compute score as

If $({}_uC_i = {}_dC_j)$ && $n \geq 2$ then

$\sum_t \mu_i = \{(n - I_r) \times (\alpha - v)\};$

Else

$\sum_t \mu_i = (\alpha - v);$

// For numeric feature compute score as

If ${}_uC_i > {}_dC_j$ then

${}_n\mu_l = \{({}_dC_j / {}_uC_i) \times v\};$

Else

${}_n\mu_i = \{({}_uC_i / {}_dC_j) \times v\};$

End for

Step 4. // Compute total similarity score of tuple as

$\sum \mu_n = {}_t\mu_i + {}_n\mu_i$

End for

Return: Aggregate Score of all $\mu_{1..k}$ tuples.

End

Figure 3.27. Feature Importance Technique (FIT) based Ranking Algorithm.

To explain the ranking methodology two case examples are considered. In the first example (Case 1) the problem of returning result sets purely based on searched parameters is explained, and in the second example (Case 2) a personalized search that can improve the quality of returned results in the form of recommendations is explained.

1. Case Examples

Case1: Consider a case example with Query 1 as below, which either returns no rows (*empty-answers*) or returns too many rows *(many-answers*) when it is relaxed.

Query1: Select AdvertisementID,Make,Bodytype,Series, Drive, Cc,Engine from CarsSpecs Where Make = 'Nissan' and Bodytype='DUAL CAB P/UP' and Series='LN167R' and Drive='4WD' and Enginesize=3.0 order by Advertisementid.

When such a query is passed to a SQL search engine, no results are returned as a car with the make of '*Nissan*' and other parameters such as body type, series, drive and engine size as searched by a user did not exist in the database. However, in cases where query relaxation/reformulation is achieved by using more flexible operators e.g. *'like'* and *'or'* and wildcards, the number of returned result sets is too high and finding the best match to a user's query is difficult. The comparative result sets retrieved using FIT is shown in Table 3.16. For the FIT algorithm the importance number (I_n) (values as shown in Table 3.15) were taken. These values of importance numbers were evaluated after analysis of the Web log data. No user profile information is used in this case. Results show that the highest scored records are those that match the user's search query most closely. The next most similar records are all ranked based on the decreasing number of parameters matched. These retrieved results and which are distinct, and most relevant to Query 1, are shown in Table 3.17.

Query 1	**No of Rows Retrieved**		
With SQL Engine	0-For Query 1 And 374482 – Query relaxation\reformulation is achieved using operators 'or', 'like', '*'.		
SQL Engine with FIT *Algorithm* (Records with top 4 Scores)	**Row count**	**Score**	**Features matching with User's Query1**
	5902	*1.0*	*4-Fully matched*
	609	0.99	3 Fully matched, 1 Partially matched
	1494	0.98	3 Fully matched, 1 Partially matched
	222	0.89	3 Fully matched

Table 3.16 An evaluation of results for query 1.

Rank	Make	Body Type	Series	Drive	CC	Engine	Score
1	Nissan	DUAL CAB P/UP	D22	4WD	2953	3.0	1
2	Nissan	DUAL CAB P/UP	D22	4WD	3153	3.2	0.99
3	Nissan	DUAL CAB P/UP	D22 SERIES 2	4WD	3275	3.3	0.98

Table 3.17 Distinct score results for search query 1 using FIT.

Clearly, in the problem of having too *many-answers*, when comparing the SQL engine with FIT top ranked result sets (highest ranked), there was a reduction of 98.42% in the total number of irrelevant records a user had to avoid searching. In the case of *empty-answers*, FIT can retrieve highly matched results and score them according to the number of features matched, whereas in the case of too *many-answers*, it will rank highly matched records higher. Thus in both cases a user has a choice of selecting top matched results as needed by him.

Case 2: This example depicts the effect of a personalized search in case of a Web database search. Returning personalized search results to visitors of a website can be achieved in multiple ways. Using FIT this can be achieved by setting the preference variable (α) of each user for highly searched features In the cases where it is not possible to determine the user's preference variable, (e.g. for users with no profile information), the importance of the number of features (I_n), which are values, based on collective user information, can be utilized.

To see how FIT can be used to give personalized search results consider an example. Two users u_1 and u_2 make the same query (Table 3.18). However, extra information contained in the user profile of user u_2 can be useful in giving personalized search results to him. The user profile shows that user u_2 has been searching for 'sedan' cars. Table 3.19 shows the score and ranking of result set for user 1 without any user profile information. The returned results for the query are re-ranked according to user u_2's interest. The score of user u_2 shown in Table 3.20 with different rankings shows how the user profile information can be helpful in returning best results.

User (u_n)	Make	Cost	User Profile Information
1	Holden	3000	Null
2	Holden	3000	Series=HG

Table 3.18 Car searched by user 1 and 2.

Car Id	Make	Series	Cost	*New Ranking*	Score
101	Holden	HT	3000	1	1
102	*Holden*	*HG*	*3012*	*2*	*0.98*
103	Holden	HQ	3012	2	0.98

Table 3.19 Top 3 Result set returned for User u_1 using FIT.

Car Id	Make	Series	Cost	*New Ranking*	Score
102	*Holden*	*HG*	*3012*	*1*	*1*
106	Holden	HG	3013	2	0.95
107	Holden	HG	3023	3	0.91

Table 3.20 Top 3 Result set returned for User u_2 using FIT and with extra user profile information.

Thus, in this case, even when two users make the same query the results displayed to them might be different. Profile information, workload information or explicit preference given by a user may be used in such cases. In the case where no such information is available, results can be returned based on the number of features matched.

3.5.3 Discussion: Ranking Methods

One of the major drawbacks of current recommender systems is the lack of ranking methodology to present results to a user. In both cases when making personalized recommendations or giving plain Web search results, the ranking method is crucial for delivering the most relevant results to a user. Once the number of recommendations to be given to a user is decided, presenting these results in an order of preference is necessary. These preferences could be derived form user's current searches, user profile, workload or number of features matched with the user's query. When delivering personalized search results in the form of recommendations to a user, all available information like profile, currently searched parameters (depicting a user's current interests) have to be kept in consideration.

The proposed FIT ranking is based on the principle of giving equal importance to all searched features as searched by a user. In the case where workload, user profile information or specific user preference is used, FIT gives the exact occurrence of these features an extra impetus relatively in consideration to other features and scores them higher. In the case where no information is available it does ranking based on

the number of features matched. Unlike previous ranking methods, it is flexible and can have a wider applicability.

Overall, the proposed ranking method can be summarized as:

1) A query ranking method for numerical, categorical and a mix of both types of attributes.

2) An effective solution for *empty-answers* and too *many-answers* problem using a global score representation method based on importance of features as searched by a user.

3) Enhancing recommendations by utilizing user profile information or user preferences especially in Web searches.

4) A global ranking method that scores highly matched results higher even in cases where there are no user profile, workload and explicit preferences specified by a user.

3.6 Summary: The Proposed Personalization Model

This chapter discussed about the various approaches, to model users search data by using model's based on two-dimensional and multi-dimensional data analysis methods. Further, the information mined from these models was used to build various user profiles. Unlike most of the previous Web personalized systems that widely use two-dimensional methods [128], [172] this thesis have used the MDD approach to model users search data for building various profiles. To build the individual profile of a user, top interests were mined from the tensor decomposed values in each dimension. These values were saved in the user profile and a recommendation strategy utilizing this information was proposed.

Clustering has been a popular approach and has been used extensively in many previous user profiling methods like [109], [189]. However, unlike most of these previous methods which cluster users interest vectors, we have first modelled all their interests as a tensor, done a dimension reduction and either extracted highest scored values from each dimension and then saved such interests in a user's profile (In case of individual profle) or done clustering on last dimensional matrix (In case of group profiles). For building group profiles, clustering on the tensor decomposed values of users dimension was done. Clustering was done to group users with similar search behaviour. Association rule mining has been used extensively in many previous Web

personalized systems to extract frequent itemsets [89], [182]. Quite similarly we have used them to derive appropriate knowledge from the large number of searches made by users. Once, users were clustered based on their search interests, association between similar users was found and unique rules that were highly preferred by the users in the group were saved in the group profile.

To cluster similar objects as searched by users two methods using tensors, where one based on object properties and the other based on ratings provided by users were proposed.

To improve the clustering results and to cluster similar objects, a FIBCLUS algorithm utilizing the Fibonacci numbers was proposed. When FIBCLUS is used for finding similarity between instances having purely numeric attributes, it reduces the similarity score of multiple attributes into a single global score. This score is then given as input to a clustering algorithm. In case of instances having mixed attributes, with both categorical as well as numeric attributes, a similarity matrix depicting similarity between instances is evaluated and given as input to a clustering algorithm.

For improving the clustering on multi-dimensional datasets, a new method DIF (Dimensional Influencing Factor) was proposed. It linearly combines various dimension values with the tensor decomposed values (obtained from various matrices after tensor decomposition) and improves the clustering by minimizing some information loss, which may have occurred during the decomposition process.

Most existing recommender systems use clustering and association rule mining, however when recommendations are made seldom any ranking methodology is ever adopted by these systems. In this thesis a ranking method that can exploit a user's profile information utilize his current search requirements has also been proposed.

Most of the existing hybrid Web personalized systems use either a mix of user profile or object profile with group profiles [10], [11], [28], [30], [36], [42], [136] however, this thesis proposes to use the three profiles, the user profile, group profile and object profile in conjunction to make best possible recomendaitons to users.

Chapter 4 discusses about pre-processing of server log data, wherein a real car sales website is taken as an example. Chapters 5, 6 and 7 deal with the evaluation methods of the proposed clustering, recommendations and ranking methods respectively.

Chapter 4 Case Study of a Sample Website

In Chapter 3, the various profile creation methods, recommendation and ranking strategies were discussed, in comparison Chapter 4, takes an analysis of a sample website and discusses how the important dimension values can be identified and filtered and then used in creating various models.

4.1 Introduction

A real website's server information is used as a prototype in this research. Some Web page categories of '*Mileage Cars*' are shown in the Figure 4.1. Users can browse information about each section of the website as desired and needed.

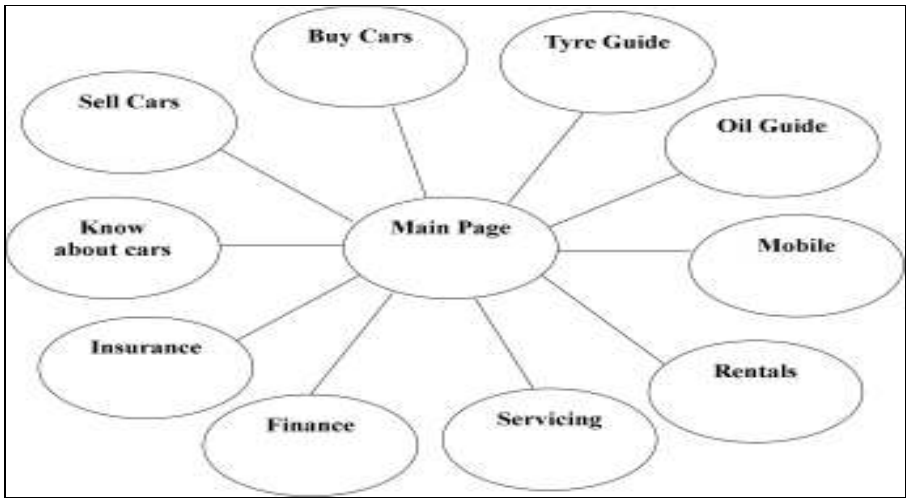

Figure 4.1 Prototype website structure of Web pages layout.

Most of the users who visit the website are either buyers or sellers. Sellers can be identified from their registration information. Such information can also be inferred from the searches of users, where sellers often explicitly look for information related to selling a car. It is easy for websites to track the sellers, as sellers place their advertisements by registering to the website. The layout of a seller's page is shown in Figure 4.2. In addition to the two categories of buyers and sellers, there is a third category of visitors, who are car enthusiast who often visit the website in order to keep their knowledge updated about new car makes and models and technical details.

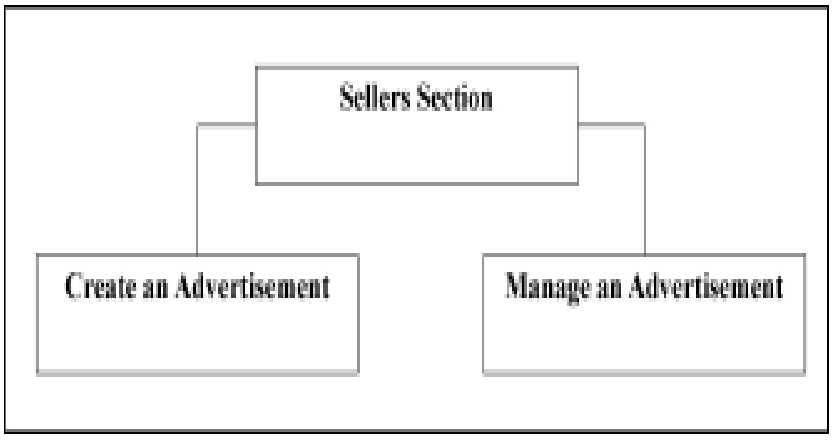

Figure 4.2 Seller's page details.

The other important feature of the website is providing detailed information about the cars available for sale to potential car buyers. There are various search parameters that potential buyers can search, including searching for a new or used car, which can then be searched according to various available makes and models, price range, location and search type.

Apart from these search parameters, some additional ones can be defined by users in the keywords option available in the interface. Some recently added new features like cars by body type, price, and age are also available for the ease of buyers For buyers of new cars the site has links to pages provided by dealers, which have bonuses or deals on them. The search interface for buyers on the prototype site is shown in Figure 4.3.

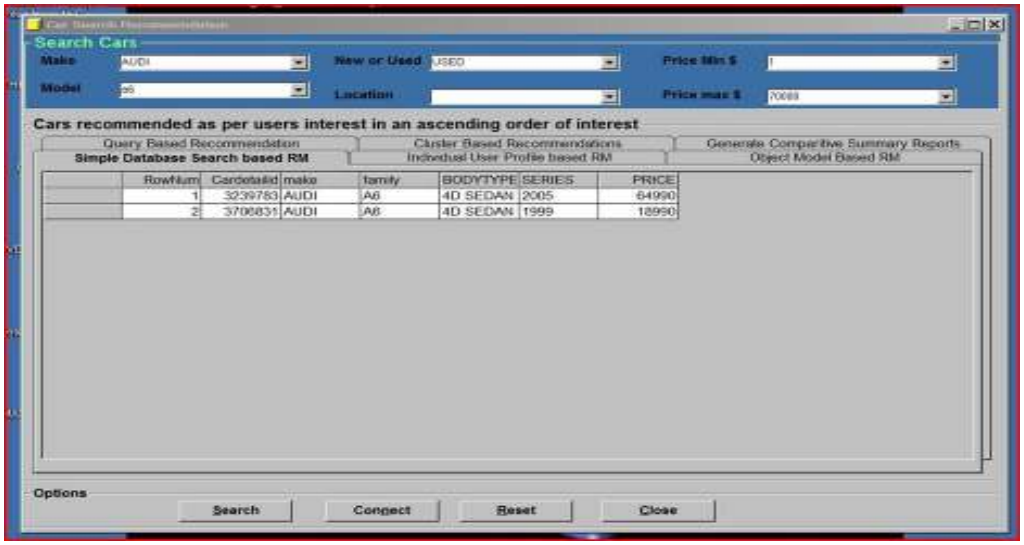

Figure 4.3 A snapshot of search interface for website '*Mileage Cars*'.

4.1.1 A preliminarily Analysis

A user's social or browsing behaviour can be of two types.

1) **Explicit** (Open): The interests of a user can be learned from his actions on the website. Examples are a). A user ranks cars based on his interests. b) He may have commented on good or bad aspects of a car in the discussion board provided by the website.

2**) Implicit** (Hidden): In the prototype website used in this research, such implicit information is located in the site sections visited, pages read, items purchased, and the searches made by a Web user, which include the details of car make, model, price range and other features.

After analysing the Web logs and website topology of the prototype website, website visitors can be divided into four categories according to their browsing trends. These categories are:

1) Potential Buyer.

2) Frequent visitor: Automobile enthusiast, dealer, distributor or agent.

3) Casual Visitor.

4) Seller.

Initial identification of user's behaviour is necessary to distinguish each category of visitor who visits the website. If buyers and sellers are clustered in the same group then recommendation may not be very effective, because these two categories of visitors have different motives. Table 4.1 summarizes the behavioural parameters, which were taken into consideration when these four profile categories were identified.

S N	Profile/ Interest Level	Description	Behaviour	Browsing Trends
1	Potential Buyer	Likely to buy a car.	Browses category and descriptions of cars he is intending to buy	Looks for categories and detailed description of a few items and compares them. Reads reviews and expert comments. Downloads information (which also helps in classification). Visits links e.g. *'First Buyers Guide'* etc
2	Frequent Visitor/ Automobile Enthusiast	Keeps himself up-to-date with industry trends	Browses randomly. Looks for car information and latest models, without much similarity in car makes and models.	Subscribes to newsletters & receives news alerts. May also be a potential buyer
3	Casual Visitor	Occasional visitor	Browses randomly. Makes 1 or 2 searches in mostly the same category (make, model and price range).	May also be a potential buyer
4	Seller (Individual or Dealer)	Wishes to sell his car	Registers and lists his car. Looks for similar cars to compare prices or looks mostly for new cars.	May also be a potential buyer

Table 4.1 Behavioural profile patterns of different categories of visitors.

Table 4.2, gives an overview of the 'body type' attribute searched by users. The valuable insights which can be gained from such data analysis is that 99.1% people search for bodytypes as specified in Table 4.2. Such analysis can identify unwanted attribute values that can be discarded from the data model.

Sno.	Car Body type	No of searches	% Total
1	NULL	48458472	89.99%
2	Sedan	1180644	2.19%
3	Wagon	1065688	1.98%
4	Hatch	1035335	1.92%
5	Ute/Pick-Up	705795	1.31%
6	Convertible	493569	0.92%
7	Coupe	381889	0.71%
8	Van	240311	0.45%
9	4WD/SUV	206853	0.38%
10	People Mover	30107	0.06%
		Grand Total	**99.91%**

Table 4.2 The top ten most popular car body types.

4.1.2 Data Filtering and Modelling

Once, preliminary analysis is done on all relevant features, pre-processing needs to be done to select essential features, clean data by removing irrelevant records and finally make this data ready for use in all experiments. Web log data pre-processing is a complex process. It can take up to 80% of the total knowledge discovery processing time [134]. The approaches adopted in this thesis for data preparation for usage and content mining include raw data cleaning, user identification, session identification, page view identification, and path completion [129]. Some of the relevant steps taken in pre-processing of raw data are discussed below:

a) **Data Acquisition:** Real server Web log data was used in all the experiments. The car sales data warehouse of '*Mileage Cars*' website had a collection of various databases. All the databases from where relevant profile building information including user searches, car details and other vital details were extracted, are shown in Table 4.3

Sno	Database Name	Size	No of Tables	Description
1	DB1	3.44 MB	9	Master Tables for visitor's responses to specific questions in the editorial section.
2	DB2	13.5 MB	14	Master Tables for visitor's subscription details and car alerts.
3	DB3	1.06 GB	85	Tables for News, Articles, Reviews on cars
4	DB4	21.5 GB	487	Website details data. Site structure database Tables, advertisements, dealer master Tables, company (service) , users promotions
5	DB5	62.2 GB	133	Statistics of website. User data, dealer data, navigational data.

Table 4.3 Server log information processed into databases.

b) **Data Cleaning, Filtering and Feature selection:** Feature selection involves identification of relevant features, file extensions and records that are vital for deriving the best available knowledge from the datasets. Let the total number of searched features be q_{n}. If q_c are the number of features which have been considered for modelling then all features discarded from the datasets are $q_{(n-c)}$. Next for each feature from $q_{1..c}$ the number of distinct feature values to be represented as dimension values are identified. Since this research uses the server log data of a car sales website, the most important features identified

were based on the users search query parameters. All datasets used for building various models, had the following five search query features: make, model, body type, cost, and car type. For each of the datasets, the number of values for each feature such as make, model, body type and cost were evaluated based on the overall size and searches done by users. E.g. for Dataset 1, the details of distinct feature values such as make and body type are shown below in Table 4.4.

Make ID	Make Name	Body Type Id	Body Group Name
1	ALFA ROMEO	1	4WD/SUV
2	BMW	2	Convertible
3	FORD	3	Coupe
4	HOLDEN	4	Hatchback
5	HONDA	5	Other
6	MAZDA	6	People Mover
7	NISSAN	7	Sedan
8	PEUGEOT	8	Ute/Pick-Up
9	RENAULT	9	Van
10	TOYOTA	10	Wagon
11	VOLKSWAGEN	11	NULL value in field
12	VOLVO		
13	NULL, No Search specified		

Table 4.4 Make and body type details for Dataset 1.

c) User- Identification: In all our experiments a user is identified by the unique IP address and the browser of his machine. Various other options like cookie deployment, customizing browser, customized programs embedded in Web pages, User's Registration information etc can be used to track the unique users.

d) Session Identification: Session refers to timestamp the user access Web pages. It can be recognized by session ID that includes a unique number which a website's server assigns a specific user for the duration of that user's visit [128]. The goal of session identification is to divide the page access of each user into individual sessions.

Once, the data is ready arranged according to user, search and time it can then be used in modelling experiments to mine valuable knowledge. All processed search log data was converted into five datasets for different experiments. The final number of attributes identified for each of the datasets from 1-5 are shown below in Table 4.5 and discussed in detail in the experiment section where they have been used. Datasets 1, 2, 3 have been used to evaluate the efficiency of clustering methods and Datasets 4

& 5 have been used to measure recommendation accuracy. These processed datasets have been used to construct various models using two-dimensional and MDD models for finding correlations between users-searches.

Category	Data set 1	Data set 2	Data set 3	Data set 4	Data set 5
Make	13	100	60	68	89
Model	38	500	455	644	667
Body type	11	16	11	12	12
Cost	14	14	14	13	13
Search Type	4	10	6	5	3
No. Of Users	25	100	10,000	949	20

Table 4.5 List of number of attributes for each dataset.

Once pre-processed data is available, relevant features as needed for group profile construction are identified. A tensor model is created of all such searches of users. As an example, in all our experiments for creating group user profiles, the five parameters used for searching like make, model, body type, cost, search type (e.g. new or used) plus a sixth dimension (users) were taken as dimensions of the tensor model [142]. The number of values for each dimension was identified by the unique number of searches for each parameter/dimension. An example of such dimension values for Dataset 4 is shown in (Table 4.6).

Dimension Name	No of unique ways or modes	Sample dimension values
Make	68	Toyota, Holden, Ford
Model	644	RAV4, Liberty.
Body Type	12	Sedan, Hatch
Search Type	5	New, Used, Ex Demo.
Cost Type	13	$1-2500
Users Id	949	1, 2,..949

Table 4.6. Number and sample of dimensions used in the tensor model.

The, overall structure of individual user tensor, consisting of five dimensions is as shown as.

$$\mathcal{T} \in \mathbb{R}^{Make \times Model \times Bodytype \times Search\ Type \times Cost\ Type}$$

Thus, the overall structure of user group tensor, consisting of six dimensions is as shown as.

$$\mathcal{T} \in \mathbb{R}^{Make \times Model \times Bodytype \times Search\ Type \times Cost\ Type \times Users}$$

For the object model when object properties are used as dimension values, the structure of the tensor model can be formed as

$$ObjectTensor(\mathcal{T}) \in \mathbb{R}^{Kms \times Transmission \times Engine \times Fuel \times Location \times Car\_Id}.$$

Conversely, when sufficient rating data is available a car model can be built as shown in Equation (12). In Equation (12), ratings are opinions given by users about certain features of an object expressed on a point scale of 1-5, where 1 is most preferred and 5 is the least preferred.

$$\mathcal{T} \in \mathbb{R}^{Ratings \times Car\ \text{Feature} \times Car\_Id}.$$

4.1.3 Summary: Case Study of the Sample Website

In this chapter a case study for the '*Mileage cars*' website is discussed. Firstly, a brief overview of the website is provided. A discussion about some category of visitors and how such visitors can be identified is discussed. Next information about segregating and analysing the category of information needed to build the particular profile category of users is discussed. To achieve this, data analysis and feature selection is discussed in brief. Feature extraction with identification of relevant feature values to be considered for modelling is also explained with an example. During the pre-processing stages the thesis has closely followed previous related works where methods of data pre-processing have been discussed [128], [129], [134]. Previously adopted popular approaches for data pre-processing (including session identification, user identification), feature selection and feature extraction have been adopted. Finally, the chapter gives a few examples of how the selected features for each category of model like model for an individual user, group user and object model can be built.

Chapter 5 The Proposed Clustering Methods: Empirical Analysis

Clustering methods are used widely when building Web personalization systems. Section 2.2.2.1 discussed about some widely adopted applicability of clustering methods when building user profiles. Similarly, Section 2.3.1.1 of the thesis detailed about some clustering methods which have been used widely for building recommender systems. This chapter discusses the various experiments conducted with various datasets to measure the quality of clustering achieved on models built from vector and tensor methods. The chapter is divided into six sub-sections. Section 5.1 details about the various datasets used in the experiments. Section 5.2 discusses about the various evaluation metrics adopted to test the clustering quality on the datasets used in various clustering experiments. Section 5.3 evaluates clustering of similar users based on tensor methods. Next section 5.4, evaluates clustering of objects based on tensor methods. Section 5.5 evaluates the performance of the proposed FIBCLUS clustering method on various datasets from the UCI repository. Finally, Section 5.6 concludes with the summary of various clustering methods as used in this research.

Once, the processed data is used for constructing various group user models (Section 3.1.3), subsequently, clustering is performed on the decomposed values. Clustering is a popular CF-based approach used to group similar users based on their search behaviour [128], [145]. For many Web service providers, individual user behaviour modelling may be a costly affair in terms of cost, time and resources. Identification of similar users and objects is a very important part of any personalization methodology and is crucial for making good quality of recommendations. The objective of clustering high-dimensional data models created from a search tensor of users is to retrieve superior grouping of users and objects. Such clusters, and the knowledge mined from clusters of similar users and objects, can then be used by Web service providers in their recommender systems for making recommendations.

5.1 Datasets Used for Measuring Clustering Methods.

To evaluate the quality of clustering given by vector-based methods, and tensor-based methods, experiments were performed on a variety of datasets. In total fifteen

datasets, belonging to two categories were used for evaluating the performance of various clustering methods. These two categories of datasets are:

1. *Web search datasets*: Under this category, five different datasets were used for measuring the performance of various clustering methods. Three real life car sales search log data, Microsoft search log data [24] and one Spam E-mail data were used in all the experiments.
2. *General Datasets*: The first dataset called *'Medical'* [116], [170] is a general dataset containing information about patients and their medical conditions. This data is multi-dimensional as many patients have one to many symptoms with each symptom represented by a range of values. This data is quite analogous to Web search data, where each visitor makes one or many searches of different objects with each object having multiple properties or dimensions. This dataset is used to evaluate the performance of clustering methods on general data that is multi-dimensional. For measuring the performance of the proposed FIBCLUS clustering algorithm, we have used various datasets from the UCI repository [56].

5.1.1 Web search datasets

In the case of *Web search datasets* the complete details of each processed Dataset 1, 2 & 3 (Section 4, Table 4.5) used in clustering experiments are discussed below.

Dataset 1: The first real-life dataset *'Dataset 1'* consists of 25 users. There were 880 searches made by these 25 users. These users had search data that is real data but synthetically divided into classes for testing purposes. Certain numbers of users showing similarity in their search behaviour were selected. This allowed us to create cluster class labels and then test the clustering methodology developed in this thesis for their performance. The groups under which these users were classified had the following characteristics:

Group 1: Users 1-4 searched for the same parameters in all searches, or all their searches were the same. E.g. all car model names, make, body type and cost ranges were same for each of these users (similar searches).

Group 2: Users 5-10 had at least two searches in common consisting of all the same search parameter, and had one search different from each other (proportionately similar searches).

Group 3: Users 11-17 had only a few search parameters like car model name, make, body type in common and other search criteria were different few parameters similar).

Group 4: Users 18-20 had no searches in common. (distinct from each other in the group and outside the group)

Group 5: Users 21-25 had missing values for search parameters, or had no values specified for certain search criteria like cost, model etc. Each of the users in this group had a distinct search from each other but the missing search parameters were prominent in each. The other noteworthy factor in this group was that each of the first four individual users in this group had two searches in common with each of the Group 1-4 respectively, but for these they had only a few search parameters in common like car make, model and the rest of the search parameters were missing. The last user in this group had no search similar to any of Groups 1-4, and all users in this group, but this user had a few missing search parameters, which was an intrinsic behaviour of this group.

Dataset 2: This dataset consisted of $290,000$ records of 100 users, which after processing, that included removing duplicate records; removing robot records; arranging similar records in a session as one record (with frequency counted); and deleting all empty records, were reduced to around $38,400$ records. There were four categories of users in these datasets, which were all selected from server logs based on the number of searches made by users. There were 25 users, who had the maximum searches, 25 who had a mean number of searches, 25 with a minimum number of searches (at least 3 searches by each user). The remaining 25 users had an average number of searches. All the users in each category were selected randomly.

Datasets with such variations were studied because of the typical characteristics of Web log data, where some users may have made many searches, while some may have just two or three searches. The other reason it was taken was to see how well a model can accommodate different users, their number of searches and still relate to similar users. These datasets were used to evaluate the performance of various clustering methods on multi-dimensional datasets.

Dataset 3: This dataset consisted of $10,000$ users. For this dataset, there were about 2.5 million records in the original dataset, which after processing were reduced to around $290,000$ searches. The mean number of search per user in this case was 29.

This datasets was used to evaluate the performance of various clustering methods on a larger scale multi-dimensional datasets.

Table 5.1 shows the typical structure of Dataset 1-3 that store users search information.

Field Name	Datatype	Description
IP	varchar(50)	IP Address of user
Id	int	IP converted to unique User ID
Ad_id	int	Unique advertisement Id
Date_time	datetime	Date- Time of search
Make	varchar(50)	Make of Car
Model/Family	varchar(50)	Model of car
Bodytype	varchar(50)	Body type category of car
Series_year	varchar(4)	Series Year of the Model
Variant	varchar(50)	Variant of the model
Price	int	Price Range
Kilometres	int	Kilometres done Range
Location	varchar(50)	Location of user
Searchtype	varchar(50)	Search type like new, used, demo.
Sessionid	varchar(50)	Identification of session id
Frequency	int	Number of time, same search done.

Table 5.1 Structure of processed data with relevant features, for Datasets 1-5.

Microsoft search data: This dataset contains details about visits to various sections (vroot) of the Microsoft website by visitors. Specific details in binary, about whether a particular section of the website is visited by a user or not are provided in the dataset. Other details like to what URL (Uniform Resource Locator) a section belongs are also provided in the dataset. The processed test data of 5000 users was taken to model the tensor. The detailed statistics of processed data are as shown below in Table 5.2.

No. of users	No. of Max Searches		Min Searches	Average Search Per User
5000	User ID	Searches	Users with only 1 Search	3
	12941	28	1547	

Table 5.2 Processed statistics of Microsoft test searches data.

E-mail Spam Dataset: This is test data extracted from three mail boxes g-mail, Hotmail and Yahoo. The information that was extracted from in-boxes of various email addresses of users, consisted of email sender, type, subject, and content of various e-mails. In cases where a sender was listed in the address book of a user he

was classified as a known sender or else was classified as an unknown sender. If the sender's address was blocked, he was classified as a blocked sender.

5.1.2 General datasets

The other category of datasets described as *general datasets* are used for measuring clustering efficiency. These datasets are investigated as they have predefined class information that can be used for measuring the quality and accuracy of various clustering methods. The two categories of datasets used in the various clustering experiments are:

1. Medical datasets: A general dataset available from [116], [170] for checking clustering performance of proposed DIF clustering method is used. The dataset contains 90 instances with 8 medical conditions and 5 measurement types for each condition. This dataset has been used in classification task for determining where patients in a post-operative recovery area should be sent next [185]. The 90 instances have been classified under 3 class categories which define what has been happening with the patients. This dataset has been taken for clustering because it has patients who have similar conditions and have been classified for the same treatment. However, in contrast to the objective of classification the primary objective is to use un-supervised learning methods to determine the appropriate cluster of users.

2. UCI Repository: Datasets from the UCI repository [56] which measure the accuracy of FIBCLUS clustering with various existing clustering methods.

5.2 Evaluation Criteria

Some popular cluster evaluation measures are *Purity*, *Inverse Purity* and their harmonic mean (*F-Score*) [12]. The inter-cluster and intra-cluster are also widely used measures for cluster evaluation. To evaluate the quality of clustering given by different methods, this thesis uses various clustering evaluation measures. These measures have been grouped under two categories, the extrinsic measures and the intrinsic measures.

1.) **Extrinsic measures**: If the similarity between two attributes a_1 & a_2 is to found using extrinsic measures, then such similarity measures take into

consideration all other attribute values in deducing the similarity between these two attributes. A simple definition of extrinsic similarity measure between two attributes as discussed by [45], [178] is given in Equation (36).

$$SA(a_1, a_2) = \sum_{a_k \in \mathbf{A}} \left| f(a_1, a_k) - f(a_2, a_k) \right| \tag{36}$$

In the Equation (36), $f(a_1, a_k)$ denotes the function that signifies the association between $a_1 \& a_k$, and $f(a_2, a_k)$ denotes the function that signifies the association between $a_2 \& a_k$. The term '*A*' denotes the set of attributes that will contribute to the calculation of extrinsic similarity between the attribute a_1 & a_2. In case extrinsic measures external labels for each instance is available, so it can be determined whether the class of the instance is same as the cluster label.

2.) **Intrinsic measures:** An intrinsic similarity measure between two instances i_j and i_k is purely defined in terms of similarity between the values of i_j and i_k. If the two instances have multiple attributes, then vectors compromising the values of i_j and i_k can be compared using various measures like Euclidian, cosine and Pearson's correlation coefficient to find the similarity between the two instances.

Table 5.3 details about the various clustering evaluation methods that have been employed on the various datasets. The tick mark symbol '√' signifies that the said evaluation method has been employed on the specific datasets and a '×' means the said measure has not been performed on the dataset.

	Clustering Evaluation Methods						
	Extrinsic Measures					Intrinsic Measures	
Dataset	**Precison**	**Recall**	**F-Score**	**Purity**	**Entropy**	**Inter-Cluster**	**Intra-Cluster**
Dataset1	√	√	√	×	×	×	×
Dataset2	√	√	√	×	×	√	×
Dataset3	×	×	×	×	×	√	×
Microsoft	√	√	√	×	×	×	×
Medical	√	√	√	×	×	×	×
E-mail	√	√	√	√	×	×	×
UCI Repository	√	√	√	√	√	×	×
Object Data-Dataset6	×	×	×	×	×	√	√

Table 5.3 Various clustering evaluation methods undertaken on different datasets.

The two measures Purity and Entropy were taken as evaluation metrics to evaluate the performance of FIBCLUS clustering on UCI datasets. These two measures are discussed briefly.

Purity: It focuses on the frequency of most common category present in the cluster [12]. Let C_k be the set of clusters, L be the set of categories or reference distribution and n be the set of clustered objects, then purity is evaluated by taking the weighted average value of maximum precision values. It is evaluated as shown in Equation (37).

$$Purity = \sum_{k} \frac{|C_k|}{n} \max \text{ Precision}(C_k, L_j) \tag{37}$$

where the precision of cluster C_k for a given category L_j is given as shown in Equation (38).

$$\text{Precision}(C_k, L_j) = \frac{|C_k \cap L_j|}{|C_k|} \tag{38}$$

Entropy: It is a measure of how the instances are distributed within a cluster. The overall entropy value is given by averaging the cluster-wise entropy values [12], and is given as shown in Equation (39). In Equation (39), $P(i, j)$ is the probability of finding an element from the category i in the cluster C_j and nj is the number of objects in cluster C_j and n is the total number of objects in the distribution.

$$\text{Entropy}=-\sum_{j}\frac{nj}{n}\sum_{i}P(i,j)\times\log_2 P(i,j). \tag{39}$$

In case of all datasets where the class definitions of clusters are well-defined, extrinsic methods can be a good measure for evaluating the performance of clustering methods. Once various clustering solutions are obtained, pre-defined cluster classes can be checked against the assigned cluster classes and efficiency of such measures can be accurately evaluated using the *F-Score*. In most of the cases since the datasets have pre-defined classes, evaluation using F-Score measure is suitable to check the quality of clustering [12]. In cases such as Dataset 3, and Dataset 6 (object data) where datasets have no pre-defined classes or where class definitions are missing, intrinsic evaluation measures are performed to evaluate the clustering quality.

Both the metrics inter-cluster similarity and intra-cluster similarity measures are used for evaluating the quality of clusters formed. A widely accepted norm for measuring cluster quality is that the lower the inter-cluster similarity, the better the clustering solutions is, and conversely, the higher the intra-cluster similarity, the better the clustering solution is.

Intra-cluster similarity: This is a measure of the compactness of elements within clusters. The higher the intra-similarity between cluster elements, the better is the clustering solution.

Inter-cluster similarity: This is a measure of how closely elements in one cluster are bound to elements in the other clusters. If each element in the two clusters is the same, it is indicated by 1, and a 0 indicates a complete dissimilarity between the elements of the clusters. Thus, the lower the inter-cluster similarity between clusters, better are the results of the clustering. Here the overall clusters are taken into account instead of singular elements. There are various methods of measuring similarity. Usually distance is taken as a criterion for measuring this similarity. Various methods based on minimum distance, maximum distance, average distance and mean distance

between the centroids of the clusters, and ward's distance-based on SSE are used to find the inter-cluster similarity between clusters. Mostly, these methods use Euclidian distance similarity measures, or cosine similarity measures. A similarity measure using Euclidian distance between two clusters C_1 and C_2 is shown in Equation (40).

$$Dist\left(C_{1n}, C_{2n}\right) = \sqrt{\sum_{n=1}^{m}(C_{1n} - C_{2n})^2} \tag{40}$$

Apart from using inter-cluster similarity measures the other measure, SSE (sum of squared Errors), is used to measure the efficiency of K-means clustering on various tensor decomposed models. SSE is defined as shown in Equation (41).

$$SSE = \sum_{i=1}^{K}\sum_{x \in C_i}(Dist(m_i, x)^2). \tag{41}$$

where x is a data point in a cluster C_i and m_i is the representative centroid for cluster C_i. K is the total number of clusters. A lower SSE usually represents good clustering. However, if the number of clusters is increased, a lower SSE may not necessarily mean good clustering.

5.3 Clustering based on Tensor Methods

To cluster similar users, the group users tensor compromising the search details of all users was decomposed. The last matrix M_n obtained after decomposition was given as an input to the clustering algorithm.

5.3.1 Experimental Design

The tensor space models were built on the user search datasets and dimensions were decided according to data features. Decomposition of the tensors using various tensor decomposition techniques like PARAFAC, Tucker and HOSVD was performed to understand the relationships between various data features. When selecting upper limit for tensor rank decomposition, this thesis has considered decompositions upto 6 tensor rank decompositions, as a higher tensor ranks decomposition may loose

valuable information and its performance becomes equivalent or worse than the two dimensional methods [98], [145]. For building group user profiles, clustering was done on the last dimensional matrix. This matrix represents user to user similarity based on the multiple searches made by users. First the EM clustering was applied on this matrix. The WEKA implementation of EM algorithm, as used in this thesis, automatically executes a 10 fold cross-validation and the log likelihood is averaged over all 10 results. In case of an increase in the log likelihood, a new clustering solution is generated and the 10 fold cross-validation begins afresh [78]. In this thesis, the 10 fold cross-validation clustering experiments were conducted with different seed values and 100 iterations of each. The log likelihood of each clustering solution was kept in consideration when choosing the best number of cluster *n*. The solution with the maximum likelihood estimate is chosen as the prefered solution. This optimal cluster number is used in guiding the optimal number of clusters in k-means and its variant algorithms. For k-means and its variants $k=n$ was considered as the optimal value of clustering methods. Once optimal clustering solutions with EM and k-means were obtained, further experiments such as deciding on the best tensor rank decomposition (Tucker-1, Tucker-2 or Tucker-3) and finding the best distance measure (Single link, Complete link, Average link or Centroid based link) were done on the selected clustering sloutions.

Datasets 1, 2 & 3: For the all the tensor-based group user models, the structure of tensor was six-dimensional and was represented using the Equation (10), Section 3.1.3 as

$$\mathcal{J}_n \in \mathbb{R}^{Make \times Model \times Bodytype \times Cost \times SearchType \times Users}.$$

Once all such group user models were created they were decomposed using the various popular tensor decomposition techniques like PARAFAC, Tucker, HOSVD, and subsequently clustering was achieved on the last matrix denoted as $\mathbf{M_n}$ of the respective tensor decomposition method.

Microsoft data: For the Microsoft data, the test 5000 users (UId) along with the sections of website (vroot) they had visited, and the respective URL (URLId) to which that section belonged were taken as the three tensor dimensions. The overall search tensor was then modelled as $\mathcal{J}_1 \in \mathbb{R}^{VrootID(293) \times URLId(294) \times UId(5000)}$.

Medical Dataset: Frequency of all records with the same class and values 1-values 8 were counted. Thus, from a total of 90 instances, the number of instances were reduced to 80. The term frequency of such instances was counted. For these 80 instances, there are 8 conditions and the possible values of measurement are from 1-5 where each 1-5 has a different meaning with respect to the individual condition. The three dimensional tensor was modelled as $\mathcal{T}_I \in \mathbb{R}^{Conditions(8)\times Measurements(5)\times Id(80)}$

Once the models were created and decomposed, they were clustered using the K-means and EM clustering algorithms [78].

E-mail Spam Dataset: Identification of spam from ham has been a challenge faced by many service providers. The concept of blocking emails from certain black listed email domains is quite feasible, but the choice of viewing information should be given to users. A certain email may be spam for one user, and for other user it may be information. At the E-mail server level, blocking or black-listing domains may be helpful in identifying a lot of potential spam. . Black-listing domains may solve the spam problem partially (e.g. as done by www.Spamhaus.org), however, similar spam E-mail disguised as a potential ham may be sent to the same user from some other domains or other email accounts.

Thus, it can be said that at an individual user level there is still a lot penetration of spam E-mails. Individual users still receive many spam emails with nearly the same content repeatedly, and each time by different users. Identification of such potential spam E-mail needs scrutiny even at the pre-delivery level at the mailbox of individual users.

A three-dimensional tensor model which is built on an individual user's preferences, and which is able to identify potential spam E-mails with a degree of certainty, is briefly discussed in this section. All E-mails with the certainty of being spam can be blocked and all E-mails whose degree is equal or lower then this level (sent by anonymous users) could be displayed in a distinctive colour. This feature with the proposed methodology can help in easier identification of such mails. Mail service providers should consider personal E-mail filters based on users mailing habits, obtained automatically. All blocked addresses in the mail box of a user can be taken as blacklisted , all other mail-id's in the address book can be taken as known senders. All E-mails from known senders should be given less spam priority.

Thus, these personal filters can combine spam filtering with other technologies, such as text classification and information filtering. The complete method of spam identification is detailed in Figure 4.1. Since this research focuses on tracking spam by its content, a tensor model is created with dimensions as $\mathcal{T}_1 \in \mathbb{R}^{Set\ of\ Keywords \times Usertype \times MailId}$.

Dimension 1: Mail with an entire set of keywords, which are in most junk mails. Each set of keywords can be assigned priority as shown in Table 5.4.

Set 1	Set 2	Set 3	Set 4	Set 5	Set 6
Lowest	Lower	Average	Medium	High	Highest

Table 5.4 Set of keywords with spam priority.

	Category 1	Category 2	Category 3
Subject	Lottery	Pharmacy	Finance
Spam	18	18	18
Ham	7	7	7

Table 5.5 Dataset categorization.

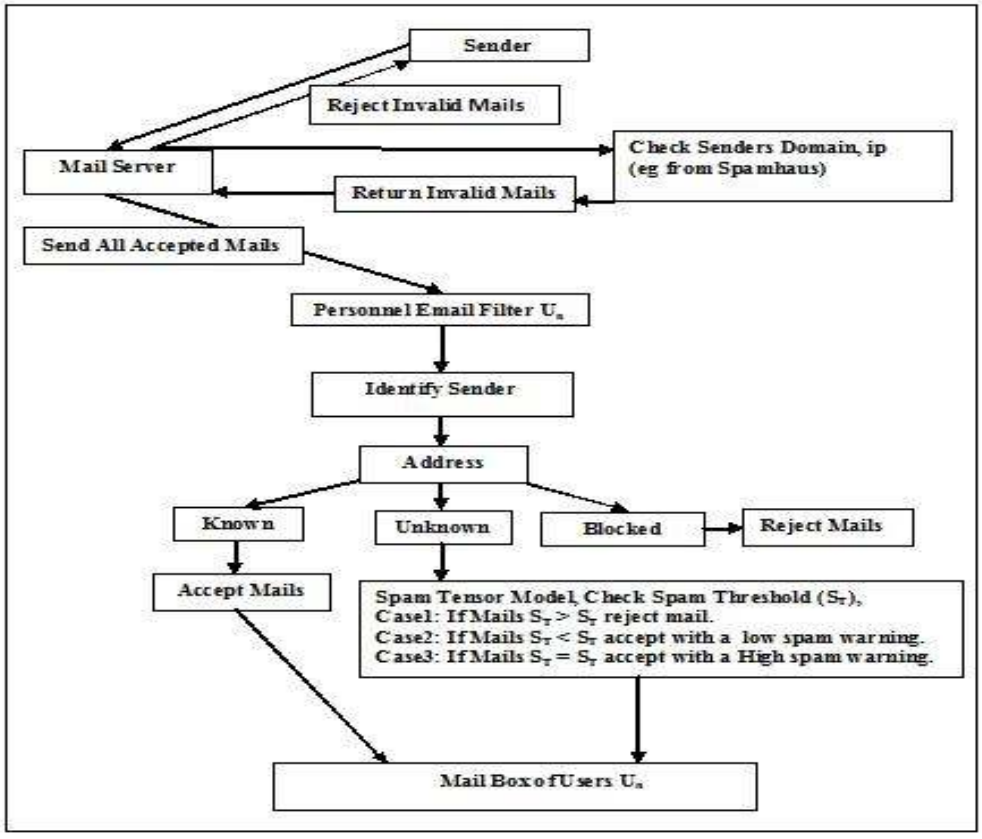

Figure 5.1 The proposed personnel spam tracker.

Dimension 2: Whether the mail is from a known sender, an unknown sender or a blocked sender. A known sender is one whose E-mail address is saved in the address book of a user.

Dimension 3: Mail Id is the incoming id of a mail.

To track spams a total of 75 E-mails from three different mail servers, with contents belonging to three identified categories as shown in Table 5.5 were considered. Thus, the tensor model created was $\mathcal{T}_1 \in \mathbb{R}^{6\times3\times75}$. All E-mails from the unknown sender have been given an extra 10% weight and all the blocked E-mails have been given twice as much weight, since prior to delivery at the personal inbox of a user, the user can be identified, and all blocked emails should be given higher spam priority. Once the spam E-mail model is created and decomposed, clustering can be done on the last component matrix obtained, to finds mails with high spam certainty.

5.3.2 Evaluation Metrics

The various evaluation metrics used for measuring the performance of clustering or for measuring the quality of clusters obtained using various methods on each datasets are explained here.

Evaluation metrics for Datasets 1 and 2: The test datasets have users that have been grouped manually based on their searches. Such identified groupings are then used as evaluation criteria for various clustering methods. Clustering achieved by various methods is compared to these identified groupings. Since, the best groupings of users are already known, utilizing this information, the widely used metrics like *Precision*, *Recall* and *F-Score* are used for evaluation. In the case of *Precision* and *Recall*, the values are evaluated based on

1. Let G_z be the set of users in each of the pre-assigned classes.
2. Let U_z be the set of users clustered in the respective cluster.

Thus for each of the clusters from, $C_{1..k}$ the value of *Precision* and *Recall* is

$$\text{Pre}\,cision = \frac{\left|G_z \bigcap U_z\right|}{\left|(G_z \bigcap U_z) + (U_z - G_z)\right|}. \tag{42}$$

$$\text{Re}call = \frac{\left|G_z \bigcap U_z\right|}{\left|(G_z \bigcap U_z) + (G_z - U_z)\right|}. \quad (43)$$

Once, the *Precision* and *Recall* values are calculated based on Equation (42) & Equation (43), *F-Score* can be evaluated. *F-Score* measures a test's accuracy and a score of 1 means best test results and a score of 0 is the worst test results. *F. Score* is calculated as shown in Equation (44).

$$F.Score = 2 \times \frac{\left(Precision \times Recall\right)}{\left(Precision + Recall\right)}. \quad (44)$$

Evaluation metrics for dataset 3: Since identification of groups is not possible due to the large number of users, and the non-availability of clear definitions, clustering efficiency for such datasets is measured by inter-cluster and intra-cluster similarity measures.

Evaluation metrics for Microsoft search data: Evaluation criteria for Microsoft data for clustering the 5000 users, using different tensor decomposition techniques is based on the sections, the title and the URL of a page visited. In total there were 110 classes into which the entire Web page URL's were classified. Examples of a few groups are shown in Table 5.6, 5.7 below.

All the users who visited the same section of a website, (i.e. title, Vroot or URL) were grouped together. Such groupings of users were evaluated based on the datasets. All users with the maximum number of similar searches were grouped together. Identification of such users was done automatically by a program, which identified best possible groupings for a user based on his search. If a user has an equal number of searches matching different clusters, such users are clustered separately. The number of such users was found to be around 5% of total users. Therefore, an error margin of 5% in optimal clustering and desired clustering has been considered while doing the evaluation. These groupings are used as a baseline to check for clustering efficiency. If users who visited the same grouped section are clustered correctly, then clustering is considered better for that clustering method. All *Precision*, *Recall* and *F-Score* values are evaluated on this basis.

SNo.	UrlId	URL
1	1052	"/word"
2	1060	"/msword"
3	1135	"/mswordsupport"

Table 5.6 An example where all URL's about MS-Word are grouped as one.

SNo.	URLid	URL
1	50	"/macoffice"
2	32	"/msoffice"
3	15	"/officefreestuff"
4	89	"/officereference"
5	222	"/msofc"
6	41	"/office"
7	234	"/off97cat"
8	220	"/macofficesupport"
9	77	"/msofficesupport"
10	193	"/offdevsupport"

Table 5.7 An Example where all URL's about Office OS are grouped as one.

Evaluation metrics for Medical dataset: The class distribution for evaluation is shown in Table 5.8 as

Class	No of Instances	Instances ID From –To Id
A	54	1-54
I	2	55,56
S	24	57-80

Table 5.8 Class Distribution.

Evaluation metrics for spam datasets: Once the tensor spam mail model is created and decomposed, the last matrix $\mathbf{M}_n$ is taken for clustering. Two clusters are formed one containing mails of known users and other containing mails from unknown users. All spam mails were scored higher due to the different set of keywords in it. The greater the number of keywords in a mail, the higher the score will be assigned to these mails. From the cluster of known users, identify mail with highest score of decomposed value and set this as the threshold limit for spam mails. E.g. if, for a cluster of a known user's mail, the highest score of any mail is 0.0004 then all mails whose score is > 0.0005 are identified as potential spam mails. The probability that a mail is spam increases with the score of its decomposed value, where mails with higher decomposed values have more probability of being spam, as they contain more keywords which are identified as being part of spam mails. Conversely, mails with lower decomposed values than the threshold values have lesser chance of being spam mails. The likelihood percentage that the mail is spam is evaluated using Equation (45) as

$$\%\ \text{Spam Certainity} = \frac{Decomposed_{value} - Threshold_{value}}{Decomposed_{value}} \times 100 \qquad (45)$$

Once the mails with high spam certainty were separated, the value of *precision*, *recall* and *F-Score* were evaluated based on the information provided in Table 5.5. The values of true positive (TP), false positive (FP) and false negative (FN) were evaluated as: all TP were spam E-mails correctly classified as spam mails; all FP were spam classified as ham & all FN were ham classified as spam.

5.3.3 Results

This section discusses the clustering results obtained with various experiments. The two fold objective of considering various experiments and different categories of datasets was:

1) To measure the efficiency of tensor models using different decomposition techniques.

2) To measure the effect of tensor best rank approximations on clustering.

Clustering results for Dataset 1: The results of clustering clearly indicate clustering superiority of TSM decomposed and subsequent clustering methods over VSM methods. Furthermore, the results in Figure 5.2-5.6 show that the proposed DIF can improve clustering of TSM models.

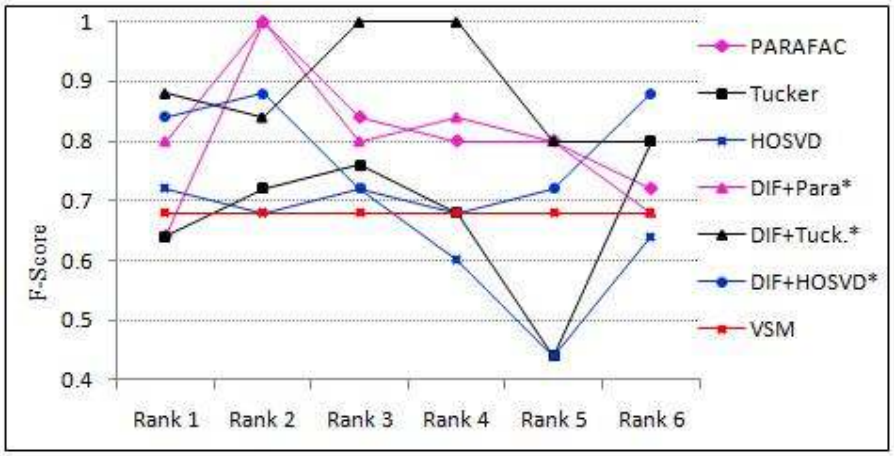

Figure 5.2 Average clustering *F-Score* of Dataset 1, where * shows the proposed methods.

For the Dataset 1 when both PARAFAC and Tucker were compared using EM and K-means and both with DIF, the results of clustering were as shown from Figures 5.3-5.6. The overall average results of *F-Score* are shown in Figure 5.2, where good performance of clustering methods is clearly visible in all the results.

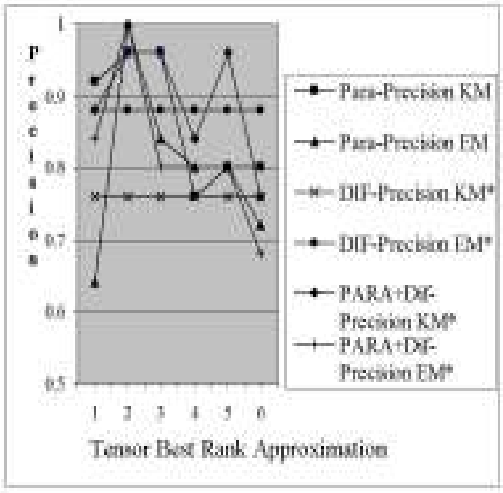

Figure 5.3 Precision results of clustering on PARAFAC, DIF for Dataset 1, where * shows proposed methods.

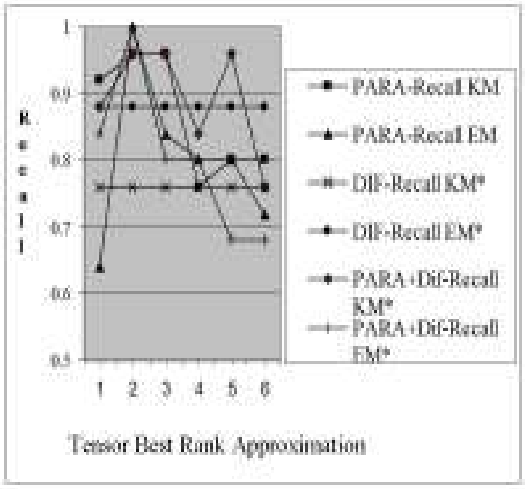

Figure 5.4 Recall results of clustering on PARAFAC, DIF for Dataset 1, where * shows proposed methods.

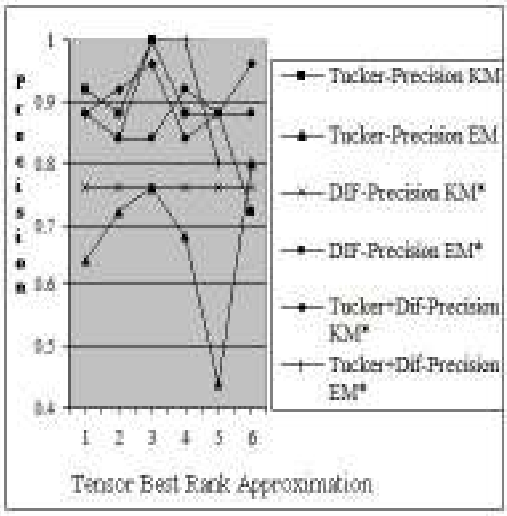

Figure 5.5 Precision results of clustering on Tucker, DIF for Dataset 1, where * shows proposed methods.

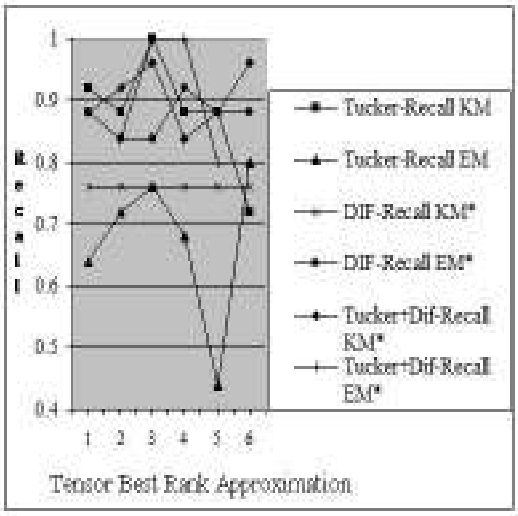

Figure 5.6 Recall results of clustering on Tucker, DIF for Dataset 1, where * shows proposed methods.

Clustering results for Dataset 2: For Dataset 2 consisting of 100 users, firstly the inter-cluster similarity results using PARAFAC and subsequent clustering methods are provided. Further to compare, the performance of TSM clustering and TSM clustering with DIF experiments with dataset 2 were conducted. The relative inter-cluster similarity values for PARAFAC, VSM and DIF clustering are shown in Figure 5.7 for the Dataset 2.

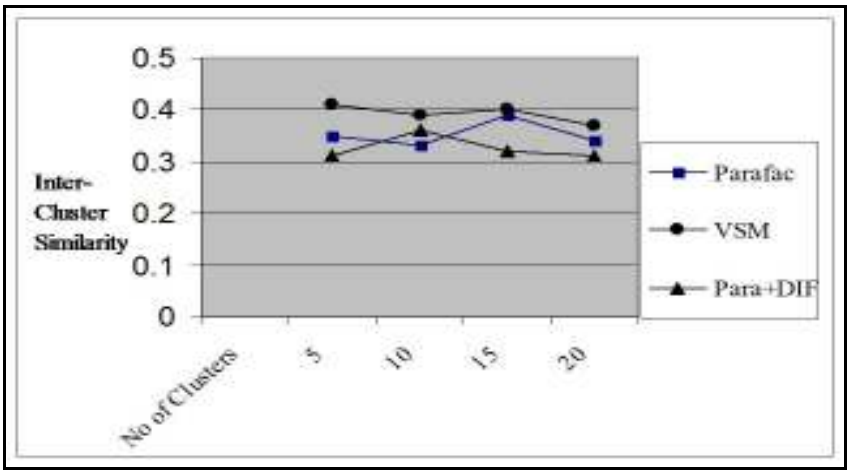

Figure 5.7 Average inter-cluster similarity using K-Means for Dataset 2.

Furthermore, to compare which TSM-based decomposition and clustering gave superior results, experiments were conducted with Dataset 2. The two popular tensor methods PARAFAC and Tucker were taken and subsequent clustering was performed on the decomposed values. To evaluate efficiency of clustering *Precision, Recall and F-Score* values were evaluated. The average results of clustering using EM and K-means were compared. Results of both clustering methods on Dataset 2 are shown in Table 5.9 and 5.10.

	PARAFAC			**Tucker**		
Rank	**P**	R	**F-Score**	**P**	**R**	**F-Score**
1	0.4490	0.1612	0.24	0.4490	0.1612	0.24
2	0.4609	0.1612	0.24	0.4637	0.1613	0.24
3	0.4508	0.1613	0.24	0.4435	0.1168	0.18
4	0.4419	0.1613	0.24	0.4467	0.1612	0.24
	Average F-Score=0.24			**Average F-Score=0.23**		

Table 5.9 Average Precision, Recall and *F-Score* for Dataset 2 using EM clustering.

	PARAFAC			**Tucker**		
Rank	**P**	R	**F-Score**	**P**	**R**	**F-Score**
1	0.4502	0.1612	0.24	0.4506	0.1613	0.24
2	0.4625	0.1612	0.24	0.4627	0.1612	0.24
3	0.4572	0.1613	0.24	0.4611	0.1612	0.24
4	0.4541	0.1613	0.24	0.4620	0.1613	0.24
	Average F-Score=0.24			**Average F-Score=0.24**		

Table 5.10 Average Precision, Recall and *F-Score* for Dataset 2 using KM clustering.

Finally, in order to see which distance-based metrics for finding inter-cluster similarity method performed the best for Dataset 2, evaluation using the inter-cluster similarity metrics based on a single link, a complete link, an average link and centroid based distance measures were used. Both EM and K-means clustering were used to find the inter-cluster similarity.

The detailed results for average EM clustering using different distance measures are shown in Figures 5.8 and 5.9, where SL is the single link, CL is complete link, AL is average link and CC is the cluster centroid. Similarly, for K-means the detailed average results of inter-cluster similarity are shown in Figures 5.10 and 5.11.

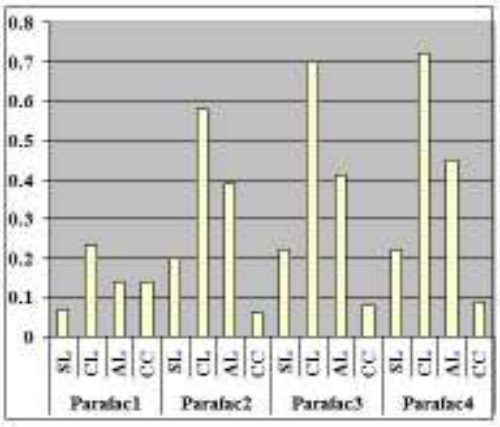

Figure 5.8. Average **i**nter-cluster similarity results for PARAFAC decomposed tensor using EM clustering.

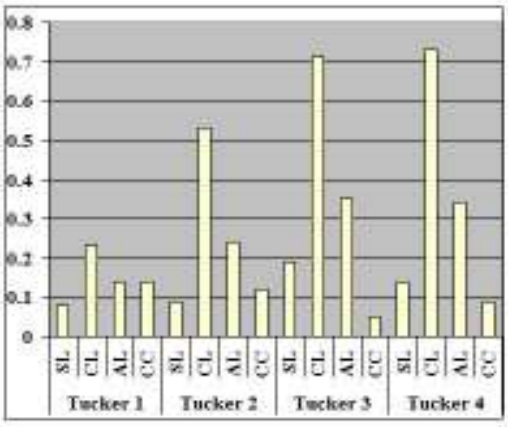

Figure 5.9. Average **i**nter-cluster similarity results for Tucker decomposed tensor using EM clustering.

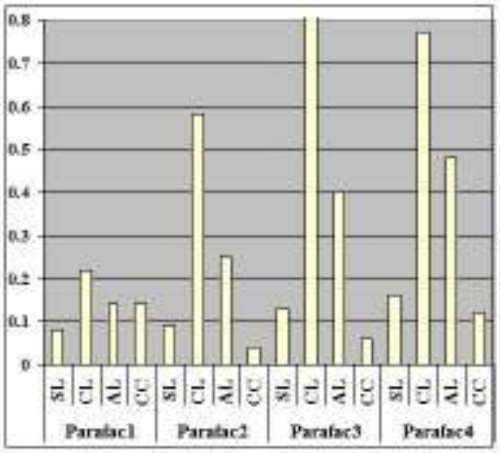

Figure 5.10. Average Inter-cluster similarity results for PARAFAC decomposed tensor using K-means clustering.

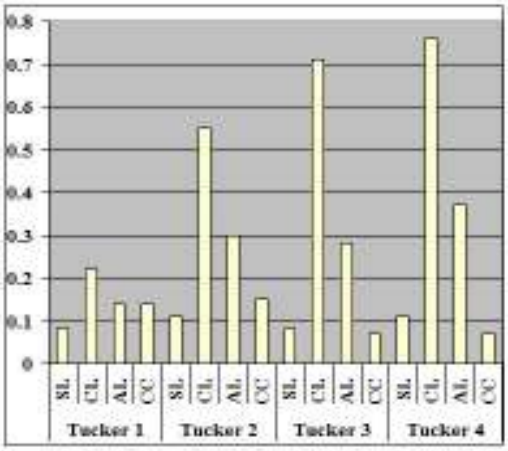

Figure 5.11. Average Inter-cluster similarity results for Tucker decomposed tensor using K-means clustering.

Clustering results for Dataset 3: For Dataset 3, consisting searches made by 10,000 users, the *Precision* and *Recall* values cannot be calculated due to the absence of ground truths and no clear class definitions of users. The average inter-cluster similarity measures are shown in Figure 5.12. It can be clearly deduced from the Figure 5.12, that the clusters given by the proposed PARAFAC and PARAFAC with DIF methods are far superior to clustering solutions given by vector-based or two-dimensional based methods.

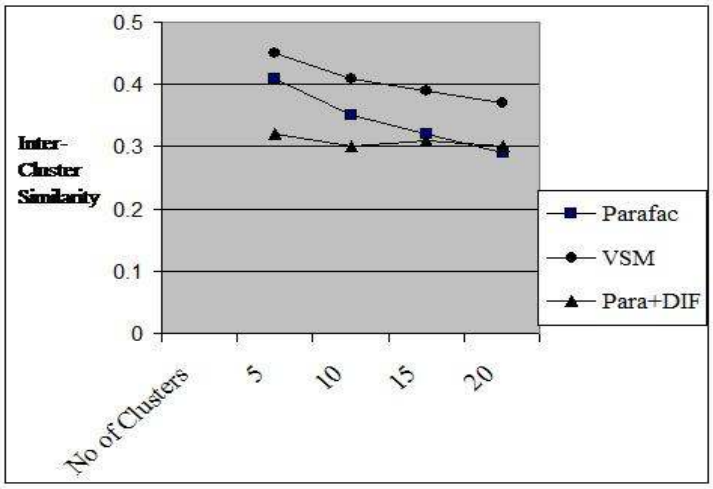

Figure 5.12. Average inter-cluster similarity using K-Means for Dataset 3.

Clustering results for Microsoft search data: Clearly, as can be seen from the results in Table 5.11 MDD modelling and subsequent tensor-based clustering out performs the traditional VSM modelling and clustering. When pre-assigning class labels to different URL's and website sections, there is a possibility that some URL's and website sections may have been wrongly classified. However, utmost care has been taken to minimize wrong classification. Even in case when a 5% error margin due to wrong classification of URL or page is assumed, still the performance of tensor clustering is still far superior to the clustering based on vector methods.

Rank	PARAFAC		Tucker		HOSVD		VSM	
	Obtained	**-5%**	**Obtained**	**-5%**	**Obtained**	**-5%**	**Obtained**	**-5%**
1	0.37	0.35	0.28	0.27	0.28	0.27	0.12	0.11
2	0.43	0.41	0.31	0.29	0.30	0.29	0.12	0.11
3	0.41	0.39	0.34	0.32	0.32	0.30	0.12	0.11
Avg.	**0.40**	**0.38**	**0.31**	**0.29**	**0.30**	**0.29**	**0.12**	**0.11**

Table 5.11 Results *F-Score* of Microsoft Data.

Clustering results for Medical Test Data: Figure 5.13 shows the performance, on general medical dataset which, again shows the TSM clustering improvement over normal VSM clustering.

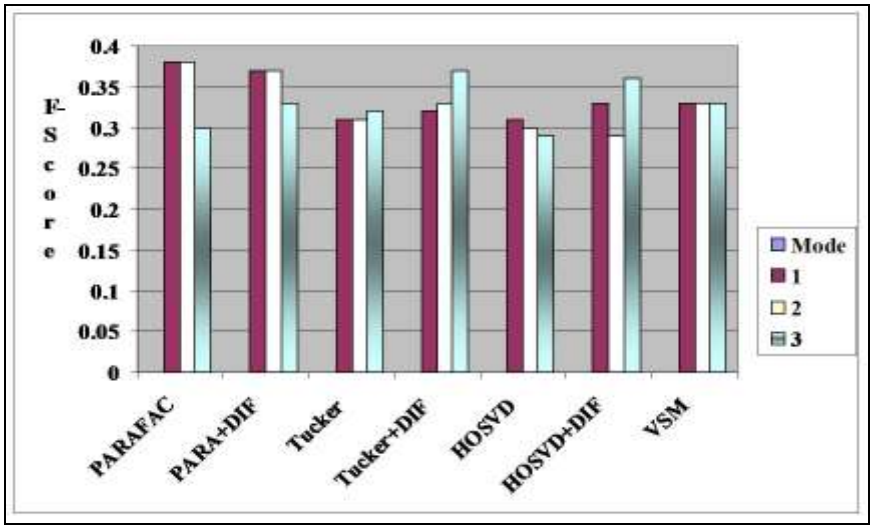

Figure 5.13 *F-Score* TSM,VSM and TSM with DIF Results of Clustering on Medical Data.

Clustering results for spam E-mails data: The F-Score details of E-mail spam datasets are provided in Table 5.12. Also as can be seen clearly from the results (Table 5.13) all the spam mails were correctly classified as spam mails with a 100% accuracy level. All mails from known senders and blocked senders were also identified 100% accurately. However, in the case of the unknown sender when the mail was not spam there was 50% accuracy. However this could have been due to the number of keywords that may have not been considered but were present in spam mails, which lowered spam certainty of these mails from unknown senders.

Technique Used	Precision	Recall	F-Score
TSM	1	0.96	0.98
VSM	0.87	0.89	0.88

Table 5.12. F-Score of clustering on E-mail spam data sets.

% Correctly classified Using method	Known Senders	Unknown sender-Not Spam	Unknown sender Spam	Blocked Sender Mails	Purity
TSM	100%	50%	100%	100%	0.96
VSM	80%	33.3%	80%	100%	0.79

Table 5.13. Results of clustering purity on E-mail spam data sets.

5.3.4 Discussion

This section discusses the clustering methods and some important conclusions drawn from the observation of multiple experiments.

a) *Impact of VSM versus TSM*: An interesting observation, whose outcome was consistent with all clustering tools and techniques used, was that TSM clustering on MDD was far superior to the normal VSM clustering. Due to the limitations of the interaction between different dimensions, two-way data analysis methods may not be suitable for multi-way data analysis. Two-way analysis methods like VSM are unable to capture the multilinear structure of Web log data as these methods suffer from rotational freedom unless specific constraints such as statistical independence, orthogonality, etc. are enforced [1].

Thus, it can be seen that the use of TSM in Web log analysis for clustering of users based on their search behaviour reveals clusters whose quality is comparatively better than that of other methods. Empirical analysis clearly shows that the proposed clustering methods based on decomposition factors of the TSM model outperformed the clustering based on traditional vector space models.

b) *Impact of various decomposition methods*: The average results of all tensor methods used such as PARAFAC, Tucker and HOSVD are quite comparable. In nearly all cases the average results given by the three methods are superior to VSM methods. When the three methods are compared the performance of Tucker and PARAFAC is quite similar, followed by HOSVD in most of the cases. Overall the average performance of Tucker methods is slightly better than the two decomposition methods.

c) *Impact of various tensor rank decompositions*: In the case of tensor decomposed models, the best results on clustering are achieved when decomposed tensors of best rank approximation equal to the mean number of dimensions/rank are taken. This can be clearly seen from Figure 5.2, (Rank 3 decomposition performs best, when there are 6 dimensions), and Table 5.8, 5.9 and Table 5.10 that the mean number of tensor rank decomposition performs best. Again in Figure 5.13 (Rank 2 decomposition performs best, when there are 3 dimensions).

One important observation which was made in all the tensor decomposition and subsequent clustering experiments was that in cases where the best rank approximation for tensor decomposition is beyond the number of dimensions, the

performance of clustering starts to decrease. At certain ranks the performance of tensor clustering methods will either be equal or be worse than the VSM models due to the complete flattening of multilinear structure. Hence, it loses valuable inter-component/factor information, thus resulting in bad clustering results.

Another interesting observation which can be inferred from Figure 5.2 is that when best rank decomposition is considered, as the rank increases the performance of plain tensor clustering methods starts to decline. In contrast, when these methods are used with DIF (Dimension Influencing Factors), the performance of clustering starts to improve. It can be clearly seen from Figure 5.2 that DIF with tensor clustering performs best when rank decomposition is of the 3rd order.

The performance of the worst performing tensor rank decomposition algorithms improves significantly when DIF is used along with PARAFAC and Tucker decomposed values for clustering. DIF is supposedly working better due to its ability to retain some extra information, which was lost during the decomposition process.

d) *Impact of clustering techniques on clustering solutions*: Another important observation is that distance-based clustering algorithms like K-means when used with DIF (Figures 5.3 -5.6) have a slight improvement in the performance, whereas density-based clustering algorithms like EM when used with DIF have a significant improvement in the clustering performance. Distance-based clustering algorithms like K-means compute the overall variance of distance between multi-dimensional attributes and cluster objects based on these, whereas density-based algorithms compute the variance in density of all multi- dimensional attributes. The similarity in output of these two clustering algorithms when combined with DIF shows that these extra attributes which are called as DIF complement effectively the overall output of clustering methods no matter what technique is used for clustering. In the case of K-means, it adds an extra distance component to the already available objects. Thus, it creates a more distinguishable distance separation between objects and thus improves the overall clustering. In the case of EM clustering it closes up the density gap between different attributes and binds objects more closely, and thus significantly improves the overall performance

e) *Impact of similarity methods adopted for clustering*: When using similarity measures between clusters, a cluster centroid seems to be the best distance measure used for measuring similarity. Overall as can be seen from Figures (CC in Figures 5.8 – Figure 5.11) average cluster centroid distance gives best solutions.

f) *Impact of clustering with DIF and clustering without DIF*: DIF is supposedly working best in most of the cases because DIF includes aggregate values derived from individual dimensions of the tensor model. When these values are used in conjunction with the tucker or PARAFAC decomposed values, any loss in the knowledge spanning all dimensions is compensated by these aggregate values to some extent. Hence, in such a scenario where the dimensions are inter-related, these extra clustering variables improve the clustering.

5.4 Evaluation of Tensor Clustering on Object Data with Features

To evaluate the clustering of objects as searched by users, this thesis clustered objects based on their features or properties. These features were the intrinsic properties of the objects, which could clearly identify the objects. Some objects had many properties similar to each other, and the whole objective of clustering was to identify those objects that were most similar to each other.

5.4.1 Experimental Design of Object Data with Features

Dataset 6: A sixth dataset called as *'Dataset 6'* was used as an object model dataset. For all experiments using an object model the cars searched by users belonging to *Dataset 4* are considered. For convenience, this dataset has been named *Dataset 6*. All distinct cars searched by the 949 users were taken for this dataset. In total 8687 cars were searched by these users. The structure of this datasets is detailed in Table 5.14.

Field Name	Datatype	Description
Ad_id	int	Unique advertisement Id
ID	int	Unique Car identification number
Make	varchar(50)	Make of Car
Model/Family	varchar(50)	Model of car
Bodytype	int	Body type category of car
Kilometres	int	Kilometres done Range
Price	money	Price Range
EngineSize	int	Engine size of car
Fuel	int	Type of fuel used by car
Colour	varchar(50)	Colour of car
Transmission	int	Transmission
Location	int	Location of car
UsedNature	char(1)	Search type like new, used, demo.
Series	varchar(50)	Series of car
Series_year	varchar(4)	Series Year of the Model
Variant	varchar(50)	Model variant
Rego	varchar(50)	Registration details
List_date	datetime	Listed on the website
Dealer_id/Seller	int	Dealer identification number

Table 5.14. Structure of Object '*Data sets 6 and 7'*.

For creating object models there can be two approaches as discussed in Section 3.1.4. The TSM model proposed for a car model in this section uses the first approach where object features data is available to create the model. Once data is processed and arranged the general structure of a tensor in this case is

$$\mathcal{T} \in \mathbb{R}^{Bodytype \times Engine\ Size \times Kms\ Done \times Transmission \times Fuel \times Car\ Id}$$

or car model with a number of ways as $\mathcal{T} \in \mathbb{R}^{12 \times 67 \times 11 \times 16 \times 12 \times 8687}$ Once the object model was created it was decomposed using PARAFAC and Tucker. Subsequently clustering using K-means was achieved to find the various clusters of similar cars.

5.4.2 Evaluation

To measure the efficiency of K-means clustering on various tensors decomposed models this thesis has used SSE. SSE measure has been already discussed in Section 5.3.2 Lower SSE usually represents good clustering, however if the number of clusters is increased, lower SSE may not necessarily mean good clustering. The other measure that has been used to measure the quality of clusters is inter-cluster and intra-cluster similarity.

5.4.3 Results

Figure 5.14 displays the SSE results of clustering on object data in comparison to VSM models. When clustering on tensor decomposition methods is compared, these results of inter-cluster and intra-cluster similarity are shown in Figures 5.15 and 5.16 respectively.

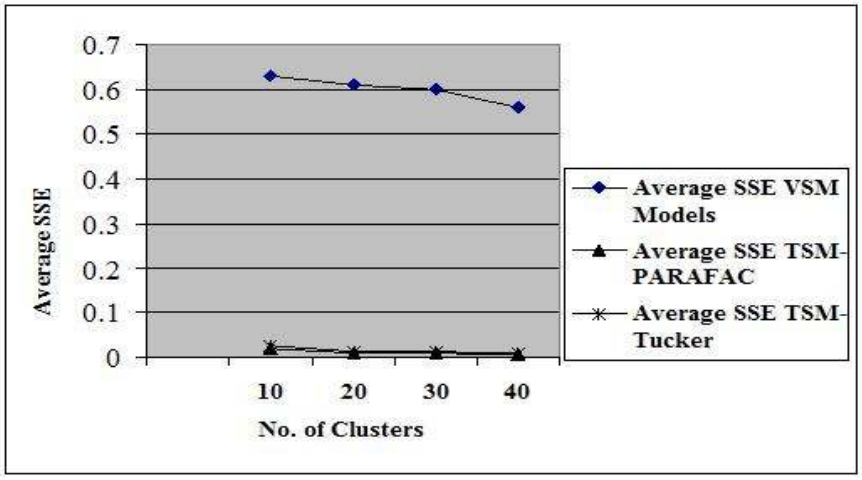

Figure 5.14 Average SSE for VSM and tensor clustering on object data.

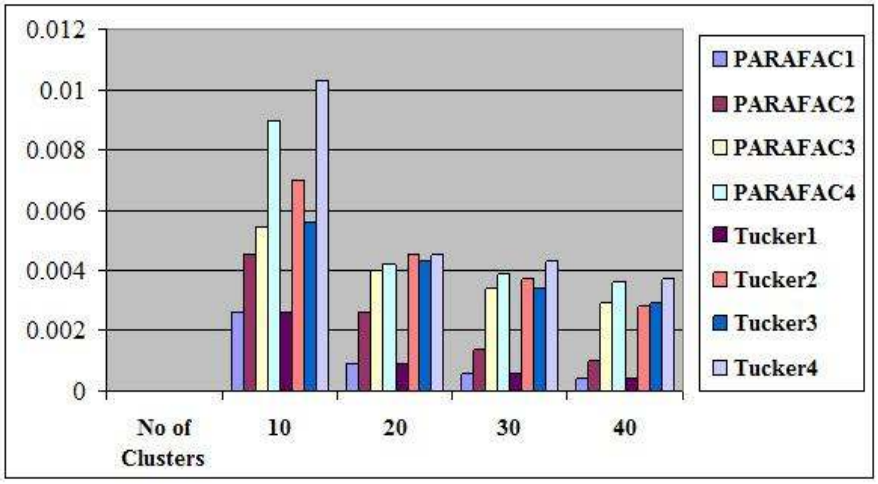

Figure 5.15 Average inter-cluster similarity for object tensor clustering.

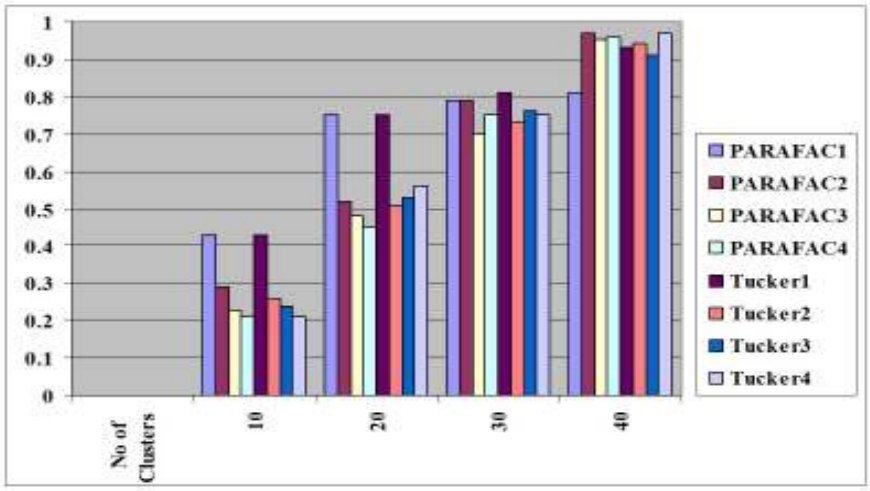

Figure 5.16 Average intra-cluster similarities for object tensor clustering.

5.4.4 Discussion

As can be observed from the results of clustering on object data (Figure 5.14) the clustering using tensors is far superior to the VSM methods. It can be seen that Tucker performance is slightly better than PARAFAC. For PARAFAC methods, the best clustering is given by rank 4 decomposition (based on inter-cluster similarity) and rank 1 decomposition (based on intra-cluster similarity). Similarly for Tucker decomposition the best clustering is given by rank 4 decomposition (based on best inter-cluster similarity) and rank 1 decomposition (based on best intra-cluster similarity).

Another interesting observation which is clearly visible from the results (Figure 5.15 and Figure 5.16) is that as the number of clusters increases, the performance of both measure starts to improve. This is a clear indicator that as the number of cluster increases, objects are more closely related to each other, rather than when they were placed in small cluster sizes.

5.5 Evaluation of FIBCLUS Clustering

The objective of the experiments was to evaluate the quality of clustering results obtained using the proposed FIBCLUS similarity scores adopted in the different clustering algorithms. Standard evaluation criteria such as Entropy, Purity and *F-Score* were used to assess the quality

5.5.1 Experimental Design for Test Datasets

Similarity scores were calculated with the proposed FIBCLUS algorithm using the Fibonacci altered values and clustered with the existing methods. For numeric datasets FIBCLUS was used with Expectation-Maximization (EM), K-means (KM) and Extended K-means (XM) [78] shown as #1, #2 ,#3 respectively in the all the results Tables 5.16, 5.17 and 5.18. Distance measure for evaluating clusters was used with numeric data because a general concept is that distance measures perform better with numeric data [77], [113], [141].

For categorical and mix data direct, repeated bi-section and agglomerative clustering methods implemented in gcluto [141] and shown as #1, #2 ,#3 in all results were used. Correlation co-efficient and cosine similarity were taken as similarity evaluation methods and the best results out of these were presented. Clustering results with FIBCLUS were compared with results using the existing methods without using FIBCLUS similarity scores.

The test datasets were obtained from the UCI repository [56], except Medical [116], [170] as detailed in Table 5.15. A total of nine datasets, three of each category were used in experiments. The primary reason for taking the UCI datasets for evaluation instead of datasets used in various other experiments was to measure the performance of FIBCLUS in an environment when the class values for each data were clearly known. The knowledge of pre-existing class values can then be used accurately for comparison with each of the clustering methods used in the experiments.

SN	Dataset	Attribute Type	No. of Attribute	No. of class	No. of instance
1	Liver	Numeric	6	2	345
2	Wine	Numeric	13	3	178
3	IRIS	Numeric	4	3	150
4	Soybean	Categorical	35	4	47
5	Balance	Categorical	4	3	625
6	SpectHeart	Categorical	22	2	267
7	Teaching	Mix	5	3	151
8	Medical	Mix	8	3	90
9	Hepatitis	Mix	19	2	155

Table 5.15 Clustering test datasets details.

5.5.2 Evaluation

All the datasets taken from UCl and the medical dataset had pre-assigned class definitions. The sole purpose of choosing such data with pre-assigned class values is to accurately measure the effectiveness of clustering methods. The pre-assigned class values of each dataset are taken as a benchmark to measure the effectiveness of proposed FIBCLUS clustering similarity method with other widely used clustering methods. The number of classes of each dataset is specified in Table 5.15.

5.5.3 Results

The complete results of clustering using K-means, X-Means, EM, direct, repeated bi-section and agglomerative with and without FIBCLUS are summarized in Tables 5.16, 5.17, 5.18. Overall, as can be seen from various evaluation metrics the performance of all clustering algorithms improves when FIBCLUS based global scores and similarity scores are used with the existing algorithms. As discussed earlier, this happens due to the separation ratio that is actively bringing similar instances together (hence, making the intra-cluster similarity larger) and separating dissimilar instances more further from each other (hence, making the inter-cluster similarity lower). Independent of the type of attributes and the clustering process used, FIBCLUS is able to produce clustering solutions of high accuracy.

	Purity of clustering results					
	Without FIB Values			**FIB Values With**		
Datasets	EM	KM	XM	#1	#2	#3
Liver	0.507	0.542	0.557	0.513	0.536	0.536
Wine	0.376	0.433	0.433	0.719	0.719	0.719
IRIS	0.907	0.887	0.880	0.960	0.960	0.960
Soybean	1.000	0.979	0.979	0.979	0.979	0.979
Balance	0.526	0.494	0.538	0.549	0.549	0.549
SpectHeart	0.528	0.614	0.614	0.772	0.772	0.772
Teaching	0.417	0.437	0.437	0.424	0.404	0.430
Medical	0.6	0.422	0.422	0.478	0.478	0.478
Hepatitis	0.516	0.542	0.542	0.638	0.638	0.652
Average	**0.597**	**0.594**	**0.60**	**0.670**	**0.670**	**0.675**

Table 5.16 Results of Purity of Clustering of all datasets.

	Entropy of clustering					
	Without FIB Values			**FIB Values With**		
Datasets	EM	KM	XM	#1	#2	#3
Liver	0.233	0.21	0.184	0.255	0.231	0.231
Wine	0.372	0.377	0.377	0.209	0.209	0.209
IRIS	0.103	0.128	0.141	0.05	0.05	0.05
Soybean	0	0.025	0.025	0.041	0.041	0.041
Balance	0.446	0.458	0.437	0.41	0.41	0.41
SpectHeart	0.195	0.188	0.188	0.012	0.012	0.012
Teaching	0.472	0.449	0.449	0.469	0.475	0.471
Medical	0.231	0.454	0.454	0.306	0.306	0.306
Hepatitis	0.300	0.297	0.297	0.225	0.225	0.221
Average	**0.261**	**0.287**	**0.284**	**0.220**	**0.218**	**0.217**

Table 5.17 Results of Entropy of Clustering of all datasets.

	***F-Score* of clustering**					
	Without FIB Values			**FIB Values With**		
Datasets	EM	KM	XM	#1	#2	#3
Liver	0.438	0.459	0.457	0.461	0.467	0.469
Wine	0.336	0.376	0.376	0.713	0.713	0.713
IRIS	0.907	0.887	0.88	0.96	0.96	0.96
Soybean	1	0.985	0.985	0.975	0.975	0.975
Balance	0.456	0.437	0.484	0.448	0.448	0.448
SpecHeart	0.517	0.422	0.422	0.436	0.436	0.436
Teaching	0.420	0.433	0.433	0.425	0.408	0.433
Medical	0.333	0.301	0.301	0.332	0.332	0.332
Hepatitis	0.437	0.467	0.467	0.595	0.595	0.606
Average	**0.538**	**0.530**	**0.534**	**0.594**	**0.593**	**0.597**

Table 5.18 Results of *F-Score* of Clustering of all datasets.

.		CLASS WISE CLUSTER DETAILS											
		Correctly Classified as per class											
Dataset	**Class Values**	**EM Co**	**EM In**	**KM Co**	**KM In**	**XM Co**	**XM In**	**FIB_EM Correct**	**FIB_EM In**	**FIB_KM Co**	**FIB_KM In**	**FIB_XM Co**	**FIB_XM In**
Liver (345)	**145**	27	52	26	39	**22**	30	35	58	31	46	31	46
	200	148	118	161	119	**170**	123	142	110	154	114	154	114
Wine (178)	**59**	44	73	13	14	13	14	**57**	3	**57**	3	**57**	3
	71	13	14	54	74	54	74	**27**	6	**27**	6	**27**	6
	48	10	24	10	13	10	13	**44**	41	**44**	41	**44**	41
IRIS (150)	**50**	50	0	50	0	50	0	**50**	0	**50**	0	**50**	0
	50	50	0	47	3	40	10	**48**	2	**48**	2	**48**	2
	50	36	14	36	14	42	8	**46**	4	**46**	4	**46**	4
Soybean (47)	**10**	**10**	0	10	0	10	0	10	0	10	0	10	0
	10	**10**	0	10	0	10	0	10	0	10	0	10	0
	10	**10**	0	10	0	10	0	9	1	9	1	9	1
	17	**17**	0	16	1	16	1	17	0	17	0	17	0
Balance (625)	**B=49;**	17	158	17	188	17	183	**9**	116	**9**	116	**9**	116
	L=288;	142	83	154	41	166	54	**169**	81	**169**	81	**169**	81
	R=288;	170	55	138	87	163	42	**165**	85	**165**	85	**165**	85
Teaching (151)	**49**	19	29	22	24	22	24	22	30	22	30	**22**	35
	50	24	21	29	50	29	50	24	24	18	16	**23**	19
	52	20	38	15	11	15	11	18	33	21	44	**20**	32
SpecHeart (267)	**0=55**	50	121	5	53	5	53	**0**	6	**0**	6	**0**	6
	1=212	91	5	159	50	159	50	**206**	55	**206**	55	**206**	55
Medical (90)	**A=64;**	**49**	20	31	16	31	16	34	10	34	10	34	10
	I=2;	**0**	7	1	16	1	16	0	21	0	21	0	21
	S=24;	**5**	9	6	20	6	20	9	16	9	16	9	16
Hepatitis (155)	**D=32;**	11	54	13	52	13	52	**0**	3	0	5	0	6
	L=123	69	21	71	19	71	19	**120**	32	118	32	117	32

Table 5.19 Detailed results of clustering (Where suffix Co=Correctly Classified and In=Incorrectly Classified, EM=Expectation-Maximization, KM=K-means, XM=Extended K-means), FIB_EM , FIB_KM and FIB_XM are EM, K-means and X-Means clustering when Fibonacci altered values are taken and clustered with the respective clustering algorithms..

Overall, the results using various evaluation metrics are summarized in Table 5.20.

Overall Percentage(%) improvements in clustering using FIBCLUS		
Similarity Measure	**Best Case-**Best of all Clustering results Versus best FIBCLUS results is taken	**Worst Case-** Worst of all clustering results versus worst FIBCLUS results is taken
Purity	12.5%	12.79%
Entropy	16.86%	23.34%
F-Score	10.97%	11.89%

Table 5.20 Summary of Results on test datasets.

In general, there was an improvement shown in all the metrics. When an average of improvements in clustering by FIBCLUS on various similarity measures considering the best improvement and worst improvement was taken, it was found out that overall there was an improvement in each type of similarity measure that was considered in experiments. There was an improvement of 12.5% in purity, 16.86% in entropy and about 10.97% in F-Score

In the case of FIBCLUS, the maximum number of categorical attributes that was used in an experiment was 35 (Soybean, Table 5.15). However, the number of categorical attributes that it can handle depends on the range of Fibonacci numbers that can be generated. This depends on the integer or float values that a programming language can store. The same is true for number of instances. Nevertheless, the overhead of creating such a matrix has to be kept in consideration.

5.5.4 Discussion

a) *Performance of FIBCLUS when clustering numeric data*: For numeric datasets FIBCLUS works reasonably well. This happens because the aggregate global score computed by FIBCLUS for each instance is able to represent all the attributes. Since each attribute is well separated by the golden ratio, the overall score of similar instances is more similar. The global score of two instances having same attributes will aggregately have the same scores. This is evident from the results. For some datasets like IRIS, unsupervised clustering using FIBCLUS is able to reach 96% accuracy which is equal to some supervised learning methods like J48 [105]. This

shows that the reduced search map obtained using the global score calculated using Fibonacci numbers is able to decrease the complexity of the grouping process.

For the Wine dataset, results are exceptionally good. The performance improvement in clustering using FIBCLUS (#1, #2, #3) is nearly 50%. For the Liver dataset, results are nearly comparable, however the clustering achieved by using it has better clusters, which is evident from the purity and entropy measures.

b) *Performance of FIBCLUS when clustering categorical and mixed data*: For the categorical and mixed datasets FIBCLUS was used with direct, repeated bi-section and agglomerative clustering algorithms [141] represented as #1, #2 ,#3 respectively. The similarity matrix given as input to these clustering algorithms manages to efficiently represent such instance similarities. For the medical dataset, EM performed quite well and better than all the other clustering algorithms like K-means and extended K-means.

c*) Effect of FIBCLUS on similarity measures*: Moreover, due to the use of Fibonacci numbers to separate the attribute values, this method enables higher intra-cluster similarity and lower inter-cluster similarity and, in hence, better clustering. Experiments with the proposed method are conducted using a total of 9 datasets, containing a mix of numeric, categorical and combinational attributes. The quality of clusters obtained is thoroughly analysed. The use of Fibonacci series assists in making the instances separable so that any clustering algorithm is able to group them effectively. The other noteworthy observation is that for all datasets the clustering solutions, obtained with FIBCLUS scores and similarity measures perform similarly. This behaviour can be seen very clearly, when each cluster is visualized for its purity in Figures 5.17-5.22. The standard EM, KM and XM methods without any space mapping derives clusters with varied purity. XM performed better than KM and EM in terms of purity metrics but the entropy given by EM was better than both. For datasets like Iris and Soybean EM performed exceptionally well when compared to a distance-based algorithm like KM and XM. However, when such datasets were used with FIBCLUS in general it was discovered that the distance-based algorithms like KM and XM performed much better than the density-based algorithm like EM. This observation indicates that FIBCLUS has the ability to improve inter-cluster and intra-cluster distances in any type of clustering method. Overall FIBCLUS gave best results in terms of all evaluation metrics used.

d**)** *Scalability of FIBCLUS*: Due to the mapping of all attributes of an instance to a global aggregate score, this method reduces the complexity inherent in the clustering process. For finding the overall similarity which is represented as a single score per instance, in the case of numeric datasets, FIBCLUS has a complexity of $O(m+n)$ where m is the number of attributes and n is the number of instances. However in most cases $m \leq n$, this complexity further reduces to $O(n)$. The complexity of instances consisting of only categorical attributes is defined as $O(\frac{n^2}{m^2}+m)$ as each instance is compared with each other instance based on the similarity score. For mixed data types the complexity is defined as shown in Equation (46).

$$O(\frac{n^2}{m^2}+n). \tag{46}$$

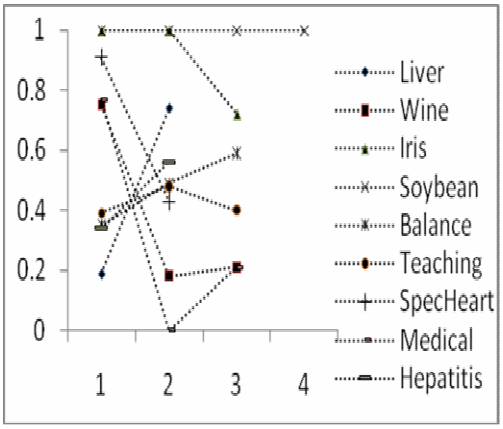

Fig. 5.17 Purity of EM clustering.

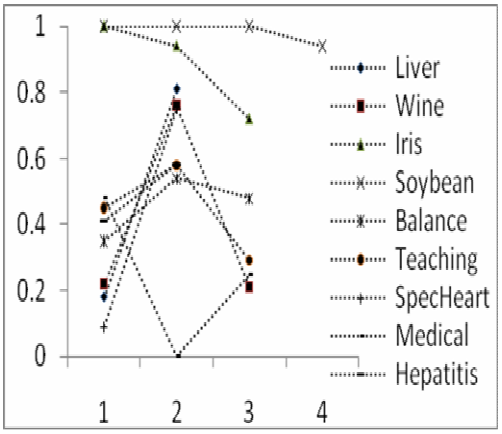

Fig. 5.18 Purity of KM clustering.

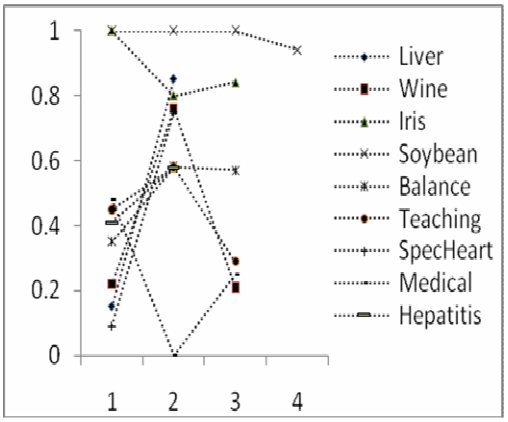

Fig. 5.19 Purity of XM (extended K-means)**.**

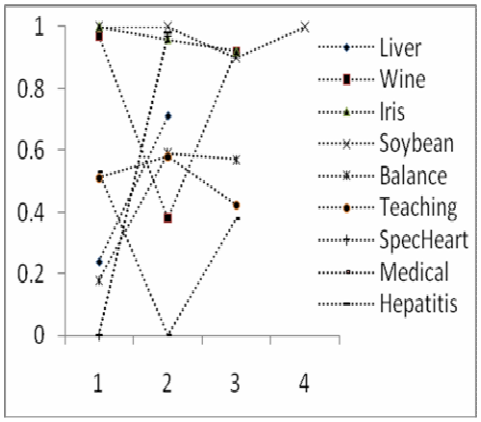

Fig. 5.20 Purity of FIBCLUS(#1 clustering).

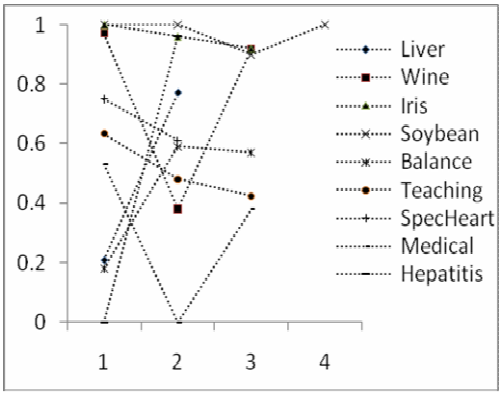

Fig. 5.21 Purity of FIBCLUS (#2 clustering).

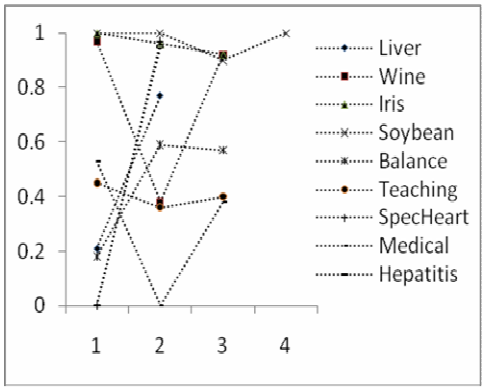

Fig 5.22 Purity of FIBCLUS(with #3 clustering).

Note: In Figures 5.20, 5.21 and 5.22 the values represented by #1, #2 and # 3 represent Expectation-Maximization (EM), K-means (KM) and Extended K-means (XM) in case of numeric datasets and direct, repeated bi-section and agglomerative clustering in case of categorical and mix datasets.

Thus in short the contributions of the proposed similarity based clustering method can be summarized as:

1) A novel clustering similarity metric that utilises the Fibonacci series to find similarities between numerical, categorical and a mix of both these data types.

2) A global score representation method for these types of attributes.

3) Enhancing existing clustering algorithms by using FIBCLUS as a similarity metrics. FIBCLUS clustering improves clustering performance. However, it cannot be used for finding similarity in the search behaviour of Web users due to multiple search instances of each user. FIBCLUS can be useful for evaluating similarity between object instances where each object is represented by a single instance, consisting of mixed attributes like categorical and numerical attributes.

5.6 Summary: Evaluation of Clustering Methods

Clustering of users and objects is an important part of the whole personalization process. In the case of clustering users, good quality of clusters ensures that a greater number of similar users are grouped in one group. In the case of objects it ensures that that a greater number of similar objects are put in one group. The key conclusions drawn based on various clustering experiments and evaluations methods are:

1. For clustering multi-dimensional Web log data containing searches of many users, two-dimensional methods used previously may not be appropriate [128],[145]. The performance of clustering on TSM is far superior to the traditional VSM based clustering.
2. Most previously used methods have clustered users interest vectors [109], [189] however, this thesis proposes to model, decompose and then use clustering on the TSM model. The research in the thesis found out that in case of TSM the performance of clustering improves when DIF is given as an extra input to the clustering algorithms.
3. Clustering of objects can improve when large numbers of clusters are formed, each containing a small number of similar objects.
4. Unlike the previously used methods of clustering categorical data based mostly on overlap similarity measures [23], [77], [81], [113] the proposed FIBCLUS algorithm enhances existing clustering techniques by lowering inter-cluster

similarity and increasing the intra-cluster similarity by transforming the attributes and creating more separable distance between dissimilar instances.

Chapter 6 Recommendation Methods: Empirical Analysis

Chapter 2.3 of the thesis discussed about the various practises adopted by researchers to build recommender systems. Section 2.3.2 gave a summary and the drawbacks of current recommender systems. Chapter 3 discussed about the various profile creation methods, how these profiles could be used for making recommendations and how the proposed FIT ranking method could be used to rank these recommendations. In Chapter 4, a case study of a car sales website called as '*Mileage Cars*' was discussed. In Chapter 5, the clustering methods used to cluster similar users and objects used for making group users and object profiles were evaluated.

This chapter discusses the evaluation of recommendation methods based on individual, group and object profiles. Making quality recommendations is the primary aim of every personalized system. Recommendations based on tensor models like PARAFAC, Tucker and HOSVD are compared with VSM based methods. To evaluate the quality of recommendations made by each of the three categories of profiles like the individual, group and object profiles using tensors, this thesis has compared the recommendations with other popularly used two-dimensional methods. Recommendations made by each of these are discussed in the following sections.

Chapter 6 is divided into five sub-sections, where first Section 6.1 evaluates the recommendations given to an individual user based on his profile. Section 6.2 measures the quality of recommendations given to group users based on their group profile. Next section 6.3 compares individual profile and group profile based recommendations. Section 6.4 evaluates recommendations given to users based on object profiles. Finally, Section 6.5 concludes with a summary of the chapter.

6.1 Recommendations Based on Individual Profiles

Evaluating the quality of recommendation for a large number of users is a costly and time consuming process. Conversely, when a small numbers of users, with different search behaviours are examined, these users's individual behaviour can be closely scrutinized. To evaluate the effectiveness of recommendations based on individual profiles built from vector and tensor methods, a fifth dataset derived from the real car sales data (Section 4.1.2, Table 4.5) called 'Dataset 5' which consisted of twenty

users, randomly selected was used. Out of these twenty users, one of the user was a frequent visitor (user 1, with 700 searches) and rest were users, that had made at least four distinct searches. The statistics of the data with the number of searches on which the user model was built is shown in Figure 6.1.

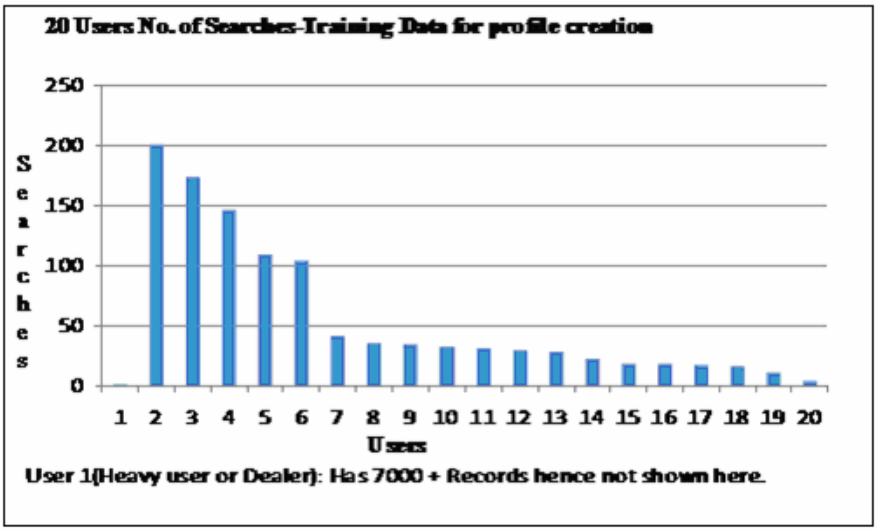

Figure 6.1 Profile creation data statistics.

6 .1.1 Experimental Design

Apart from building individual tensor models for each user as discussed in Section 3.1.2, various other models like models based on frequency, association rule mining and two-dimensional methods like SVD, PCA and NNMF were built based on the 'Dataset 5'. For each of these methods used, the make and model of a car are taken as components to be analysed as all cars can clearly be identified from these two components/dimensions. The various individual profile building methods that were used are

a) Frequency based (Top Υ objects as searched by each user are recommended).
b) Association (Important searched components like make, model from each user's search are grouped individually to find associations, then only unique rules are recommended from these associations as explained in Section 3.3.2.
c) For two-dimensional methods like SVD, PCA and NNMF, a search matrix for each user searches was created as seen in Section 3.3.1. Next, these matrices were decomposed to find top rated features. The *top* Υ make and model of cars

were then saved in the user's individual profile and were given as recommendations.

6 .1.2 Evaluation methods

The performance of recommendations based on tensor methods was compared with various other methods such as those based on frequency, associations of searches, SVD, PCA and NNMF. To evaluate the quality of recommendations of various methods, the subsequent searches of each user were taken. These searches of each user were taken after the individual profile model of each user was created. This data is taken to map the recommendations provided by tensor, frequency, association and two-dimensional profile building methods. The statistics of the evaluation data is shown is Figure 6.2, and how various evaluation metrics are derived from it are discussed below.

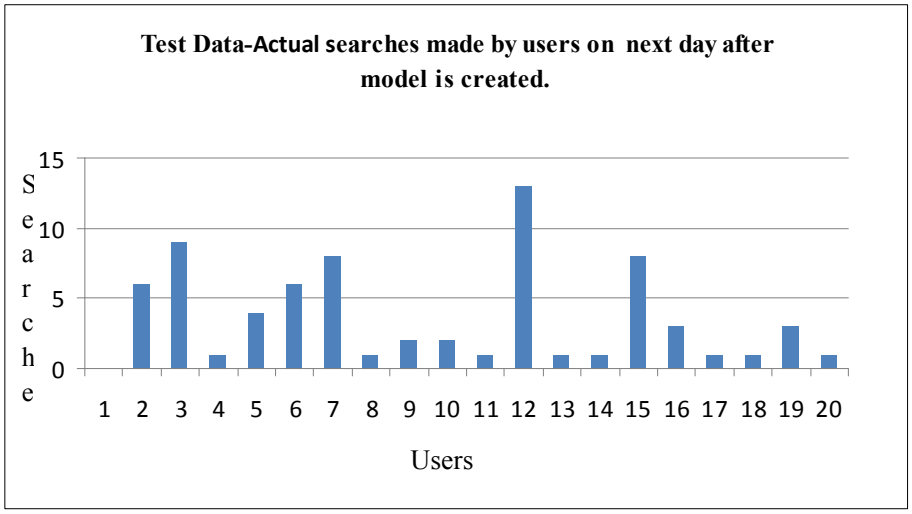

Figure 6.2 Users searches against which recommendations are measured.

Precision, *Recall* and *F-Score* are used to measure the accuracy of recommendations, where each is defined as: let $\Re$ be set of recommendations made by the model and let '*S'* be the set of actual searches done by the user, then

TN/True Negative: Set of recommendations ($\Re$) which, did not belong to user's actual searches (searches done, after model was created) and were not recommended.

$$TN = \left|(\Re - S) \cap S\right| \tag{47}$$

TP/True Positive: Set of recommendations $(\Re)$ which belong to searches (S) and were correctly recommended.

$$TP = \left|(\Re \cap S)\right| \tag{48}$$

FN/False Negative: Set of searches (S) that were not recommended $(\Re)$.

$$FN = \left|(S - \Re)\right| \tag{49}$$

FP/False Positive: Incorrect recommendations, meaning recommendations $(\Re)$ which did not $\in$ to searches (S) were recommended.

$$FP = \left|(\Re - S)\right| \tag{50}$$

Thus, *Precision* and *Recall* were evaluated as

$$\text{Pre}\,cision = \frac{\left|(\Re \cap S)\right|}{\left|(\Re \cap S) + (\Re - S)\right|}. \qquad \text{Re}\,call = \frac{\left|(\Re \cap S)\right|}{\left|(\Re \cap S) + (S - \Re)\right|}.$$

And finally *F-Score* as

$$\text{F-Score} = \frac{2(Precision \times Recall)}{(Precision + Recall)}; \tag{51}$$

6.1.3 Results

The average individual profile based recommendation results of all users by different methods are shown in Table 6.1. The proposed tensor profile based recommendation methods are denoted by PARAFAC, Tucker and HOSVD, and each numbered suffix e.g. 1, 2 and 3 at the end of these, represents the tensor rank decomposition of the respective tensor using PARAFAC, Tucker and HOSVD methods.

Average Precision-Recall of All Methods								
Method	**Top3**		**Top5**		**Top10**		**Top15**	
	Pr	**Re**	**Pr**	**Re**	**Pr**	**Re**	**Pr**	**Re**
PARAFAC1	0.41	0.63	0.23	0.64	0.16	0.60	0.20	0.58
PARAFAC2	0.40	0.53	0.22	0.54	0.15	0.52	0.13	0.33
PARAFAC3	0.38	0.62	0.23	0.68	0.20	0.69	0.22	0.51
Tucker1	0.38	0.61	0.20	0.57	0.13	0.50	0.11	0.45
Tucker2	0.37	0.57	0.19	0.51	0.12	0.49	0.09	0.46
Tucker3	0.42	0.64	0.21	0.57	0.13	0.56	0.10	0.54
HOSVD1	0.38	0.61	0.20	0.55	0.13	0.54	0.11	0.54
HOSVD2	0.38	0.62	0.19	0.55	0.12	0.53	0.09	0.51
HOSVD3	0.39	0.63	0.21	0.57	0.15	0.60	0.12	0.62
Frequency	0.30	0.62	0.21	0.62	0.14	0.54	0.13	0.40
Association	0.37	0.63	0.24	0.61	0.16	0.59	0.10	0.36
SVD	0.35	0.54	0.18	0.55	0.13	0.59	0.09	0.54
PCA	0.27	0.36	0.12	0.29	0.07	0.31	0.06	0.32
NNMF-1	0.35	0.54	0.17	0.51	0.13	0.59	0.09	0.54
NNMF-2	0.38	0.63	0.19	0.58	0.14	0.57	0.11	0.37
NNMF-3	0.37	0.67	0.18	0.63	0.11	0.62	0.10	0.48

Table 6.1 Average Precision and Recall of all methods.

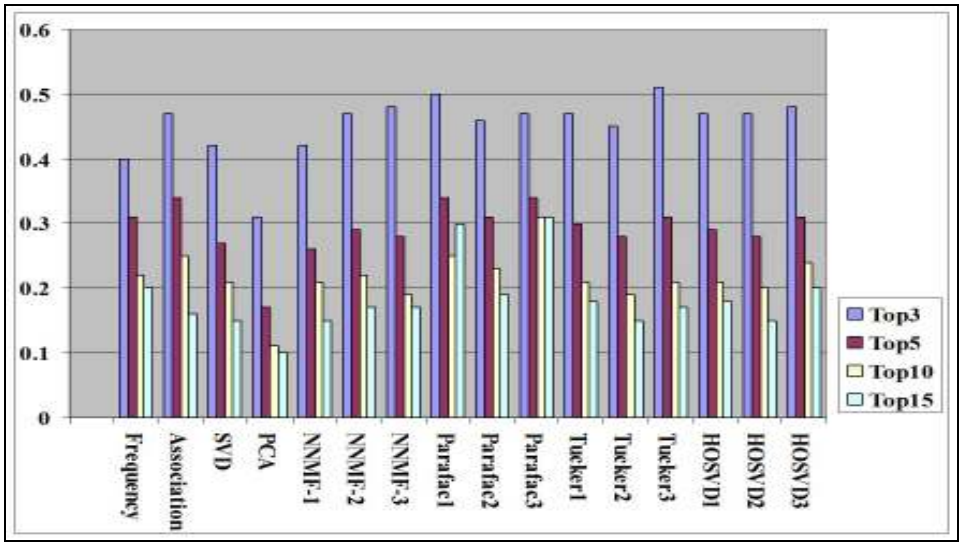

Figure 6.3 *F-Score* of all methods used for recommending to individual user.

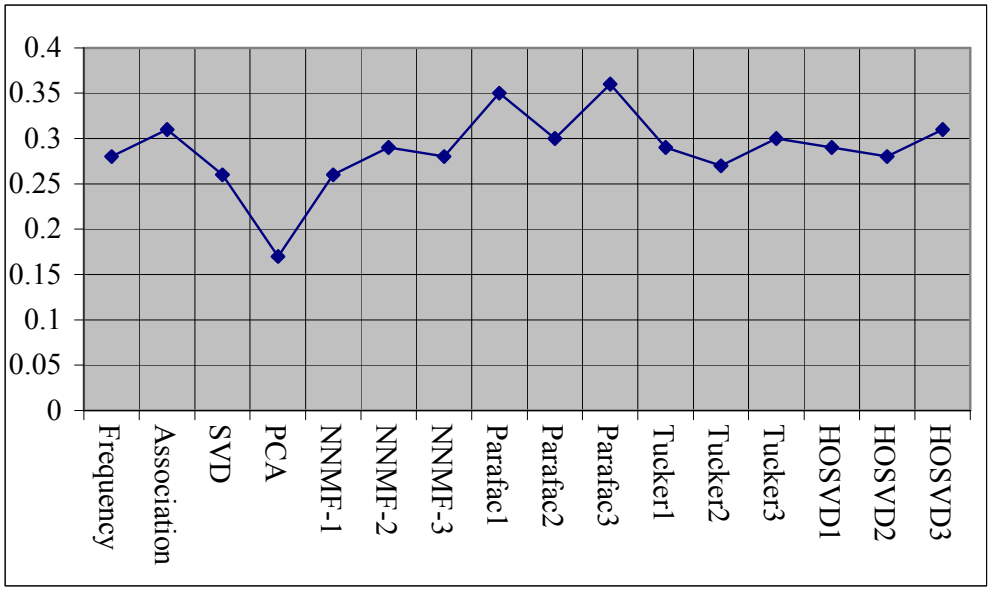

Figure 6.4 Average *F-Score* (Top3-Top15) of all methods used for recommending to individual user.

Figure 6.3 gives the comparative *F-Score* results of recommendations of each method for the top 3-top 15 recommendations and Figure 6.4 gives the overall average score of these methods. PCA performs the worst in most of the cases followed by SVD. In the case of two-dimensional methods, only NNMF performance was relatively comparable to tensor methods. Since NNMF performance was best in terms of matrix methods, a further comparison of *F-Score* results of NNMF with TSM were considered. These results are displayed in Table 6.2 and 6.3.

	Method	Top3	Top5	Top10	Top15
	NNMF-1	0.42	0.26	0.21	0.15
	NNMF-2	0.47	0.29	0.22	0.17
	NNMF-3	0.48	0.28	0.19	0.17
Average		**0.46**	**0.28**	**0.21**	**0.16**
	PARAFAC1	0.50	0.34	0.25	0.30
	PARAFAC2	0.46	0.31	0.23	0.19
	PARAFAC3	0.47	0.34	0.31	0.31
Average		**0.48**	**0.33**	**0.26**	**0.27**
	Tucker1	0.47	0.3	0.21	0.18
	Tucker2	0.45	0.28	0.19	0.15
	Tucker3	0.51	0.31	0.21	0.17
Average		**0.48**	**0.30**	**0.20**	**0.17**
	HOSVD1	0.47	0.29	0.21	0.18
	HOSVD2	0.47	0.28	0.20	0.15
	HOSVD3	0.48	0.31	0.24	0.20
Average		**0.47**	**0.29**	**0.22**	**0.18**

Table 6.2 Comparative average *F-Score* results of TSM and NNMF.

Methods	Top 3	Top 5	Top 10	Top 15
NNMF 1-3	0.46	0.28	0.21	0.16
TSM (PARAFAC 1-3, Tucker 1-3, Hosvd 1-3)	0.48	0.31	0.23	0.21
% Improvement	4.35 %	10.71 %	9.52 %	31.25%

Table 6.3 Average summary of results of TSM and NNMF.

Table 6.4 shows the average sum of matrix-based methods in comparison to the tensor-based methods. It can be clearly seen from these results that overall tensor-based methods perform much better than the matrix and vector-based methods. For the top most recommendations, which is the top three recommendations, the overall performance improvement was around 20%, and for the maximum number of recommendations, such as the top 15, the overall improvement in the recommendations was around 50%.

Methods	Top 3	Top 5	Top 10	Top 15
PCA,SVD, NNMF 1-3	0.40	0.24	0.18	0.14
TSM	0.48	0.31	0.23	0.21
% Improvement	20%	29.18%	27.78%	50%

Table 6.4 Average Summary of *F-Score* results of matrix methods and TSM based methods.

In cases where the average of all methods used for making recommendations to individual user's were compared with the proposed tensor-based methods, it can be clearly seen from Figure 6.5 that tensor-based based methods perform reasonably well for all top 3 to top 15 recommendations.

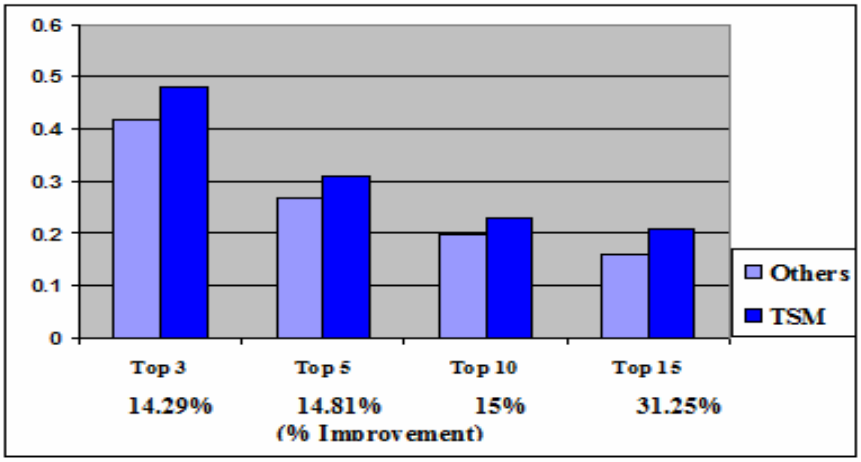

Figure 6.5 Average summary of *F-Score* results of matrix methods and TSM based methods.

6.1.4 Discussion

Clearly, as can be seen from the results, the proposed TSM based recommendation outperforms the vector and matrix-based recommendation methods. The recommendations given using unique rules, mined from association rules, performed well on average because in the cases where only a few searches were made by users, there were fewer unique rules generated, and any matching of these rules with users actual searches resulted in higher values of *Precision* and *Recall*. In cases where the recommendations were done using association rule methods, longer rules or rules based on large number of item sets were rare thus decreasing the performance of recommendations. On the other hand, when rules based on a small number of item sets are considered, too many rules with similar score are generated making it difficult to choose the best rules.

In terms of matrix methods, NNMF is the only method that performs closer to the TSM based recommendations. This is because NNMF is analogous to tensors in two-dimensional spaces because it produces non-negative vectors that are not necessarily orthogonal vectors in two-dimensional spaces. All the other methods like SVD, PCA have considerably lower performance than the NNMF and tensor methods.

Frequency based recommendations can be biased towards highly searched items and association based recommendations need a considerable amount of data to generate good unique rules.

6.2 Recommendations Based on Group Profiles

To measure the quality of recommendations based on group profiles this thesis used Dataset 4 (Section 4, Table 4.5). This dataset had search and '*leads*' details of users. A '*Lead*' is an inquiry or a requisition of details for a specific product made by an online visitor who is interested about the advertised product. Usually, leads are made by website visitors when they are interested to buy a product, and to do so they fill out the relevant (contact/email us) information in a website. This dataset consists of 949 Users. These users have been chosen based on the searches and subsequent leads made after browsing the website. Some statistics for the user searches and leads are shown below in Table 6.5.

No. of Sessions	No. of Users	Average searches per user	No. of Leads	Avg. Lead/User
2692	949	14	1649	1.74

Table 6.5 Statistics of filtered data used for tensor modelling.

6 .2.1 Experimental Design

Once group user profiles are created using tensors (Section 3.3.2) and the decomposed values of users dimensions are clustered, then all searches made by users in a cluster are grouped and frequent associations based on desired query components (like make/model) are mined. Thus, each cluster contains the searched parameters, as searched by users of the respective cluster. Association rules are mined from each individual cluster. From these association rules unique rules are saved in group user profiles. The *top* Υ rules are then given as recommendations to each user in a similar cluster.

6 .2.2 Evaluation Method

By analysing the leads and subsequent searches of users, it can be deduced that whether there are any changes in the interests of users. In all the experiments, the data used for making recommendations were taken prior to a user making a lead or inquiry about the object. The major objective was to match leads with the correctly made recommendations by various traditional CF methods and CF methods based on the tensor model. To compare the recommendations made by tensor methods, recommendations based on two-dimensional methods like Euclidian and cosine similarity are evaluated.

To evaluate the efficiency of top Υ recommendations given by each method the following metrics based on *Precision* and *Recall* values was considered. Let L_z be the set of leads made by a user u_z, and let $\Re_z^{\Upsilon}$, be the set of *top* Υ recommendations given by various methods to u_z, where $\Upsilon \geq 3$ *and* $\Upsilon \leq 15, \Upsilon \in \{\{3\},\{5\},\{10\},\{15\}\}$. Then the number of leads matching recommendation is given as $\Re_z^{\Upsilon} \bigcap L_z$. Finally the *Precision* (Pr_n) and *Recall* (Re_n) for each user u_z is evaluated as

$$Pr_z = \frac{\left|\Re_z^{\Upsilon} \bigcap L_z\right|}{\left|(\Re_z^{\Upsilon} \bigcap L_z) + (\Re_z^{\Upsilon} - L_z)\right|} \tag{52}$$

$$Re_z = \frac{\left|\Re_z^{\Upsilon} \bigcap L_z\right|}{\left|(\Re_z^{\Upsilon} \bigcap L_z) + (L_z - \Re_z^{\Upsilon})\right|} \tag{53}$$

6.2.3 Results

The average details of recommendation results for each method are shown below in Table 6.6 and the overall average *F-Score* for all top 3-15 recommendations are shown in Figure 6.6. In Table 6.6 and Figure 6.6 the acronyms used are U-ESM (Users-users Euclidian Similarity Measure) and U-CSM (Users-users Cosine Similarity Measure). U-ESM and U-CSM are evaluated based on users-items relationships, where such relationships are represented as vectors in two-dimensional spaces. Clustering is done on these vectors to find users with similar interests. These two methods adopt a CF and a content-based approach using different similarity measures. Here, the content or searches made by users is used to compare and group similar users. P1-EM, P2-EM and P3-EM are the PARAFAC best rank approximation in the order 1, 2, 3 respectively, with clustering achieved using Expectation-Maximization (EM) [79]. Similarly P1-kM, P2-kM and P3-kM are the PARAFAC best rank approximation of order 1, 2, 3 respectively, with clustering achieved using K-means. For tensor methods, mostly PARAFAC modelling is taken because the overall performance of both methods the PARAFAC and Tucker methods was almost identical/nearly the same in most of the experiments. HOSVD was not taken because in nearly all the cases the average performance of it was below these two methods. In all the Tables from 6.6-6.14 showing various recommendation results, 'Avg.' means the average score for the *top* Υ recommendations. The other values in these Tables are Pr=Precision, Re=Recall and Fs=F-Score. The Tables from 6.7-6.14 further show the results in details for each method adopted. In these Tables CL=10, CL=20 and CL=30 shows recommendation scores when the number of clusters is 10, 20 and 30 respectively.

	Top 3 Recommendation			Top 5 Recommendation			Top 10 Recommendation			Top 15 Recommendation		
	Pr	Re	Fs	Pr	Re	Fs	Pr	Re	Fs	Pr	Re	Fs
U-ESM	0.08	0.10	0.06	0.06	0.12	0.06	0.07	0.16	0.06	0.07	0.16	0.06
U-CSM	0.24	0.46	0.31	0.23	0.56	0.31	0.22	0.59	0.30	0.22	0.68	0.29
P1-EM	**0.35**	**0.60**	**0.43**	**0.32**	**0.59**	**0.40**	0.22	0.67	0.30	0.21	0.69	0.28
P2-EM	0.25	0.38	0.25	0.24	0.4	0.25	0.17	0.55	0.18	0.16	0.61	0.18
P3-EM	0.34	0.56	0.40	0.29	0.57	0.36	**0.24**	**0.69**	**0.29**	**0.23**	**0.72**	**0.28**
P1-kM	0.15	0.25	0.17	0.11	0.26	0.15	0.13	0.53	0.21	0.11	0.57	0.17
P2-kM	0.22	0.47	0.29	0.21	0.51	0.27	0.14	0.65	0.22	0.12	0.73	0.19
P3-kM	0.22	0.44	0.28	0.19	0.48	0.25	0.16	0.63	0.24	0.14	0.67	0.21

Table 6.6 Score of Top 3-15 recommendations for various methods.

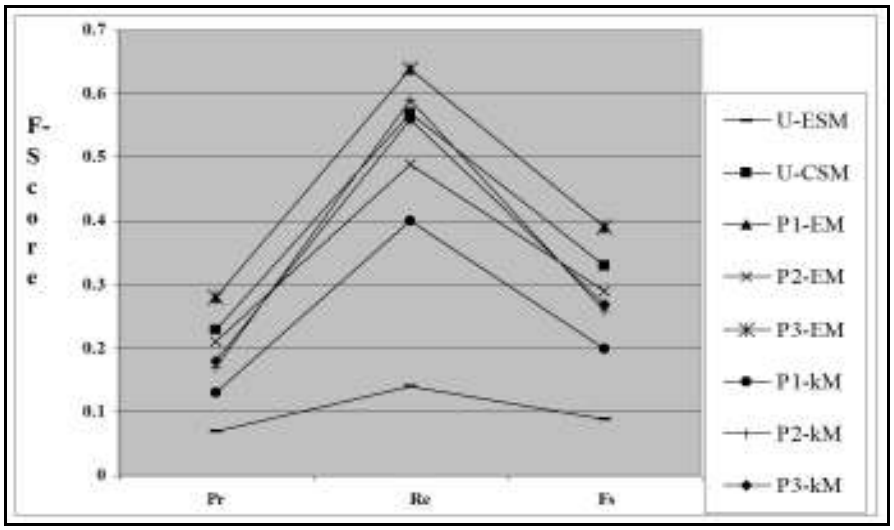

Figure 6.6 Average *F-Score* for all Top 3-15 recommendations of various methods.

Avg. CF-based recommendation- User to User Similarity (Euclidian Distance)- U-ESM												
	Top 3 Recommendation			**Top 5 Recommendation**			**Top 10 Recommendation**			**Top 15 Recommendation**		
	Pr	**Re**	**Fs**	**Pr**	**Re**	**Fs**	**Pr**	**Re**	**Fs**	**Pr**	**Re**	**Fs**
CL=10	0.07	0.11	0.08	0.05	0.14	0.07	0.05	0.19	0.07	0.07	0.11	0.08
CL=20	0.08	0.04	0.02	0.08	0.03	0.02	0.11	0.03	0.02	0.08	0.04	0.02
CL=30	0.08	0.14	0.09	0.06	0.18	0.09	0.06	0.25	0.09	0.05	0.33	0.09
Avg.	**0.08**	**0.1**	**0.06**	**0.06**	**0.12**	**0.06**	**0.07**	**0.16**	**0.06**	**0.07**	**0.16**	**0.06**

Table 6.7 Average Precision, Recall and *F-Score* for CF-based methods using Euclidian similarity methods.

Avg CF-based recommendation- User to User Similarity (Cosine)- U-CSM												
	Top 3 Recommendation			Top 5 Recommendation			Top 10 Recommendation			Top 15 Recommendation		
	Pr	Re	Fs	Pr	Re	Fs	Pr	Re	Fs	Pr	Re	Fs
CL=10	0.29	0.49	0.35	0.29	0.60	0.37	0.28	0.6	0.35	0.27	0.67	0.34
CL=20	0.23	0.48	0.31	0.21	0.55	0.29	0.20	0.62	0.28	0.20	0.72	0.29
CL=30	0.20	0.41	0.26	0.21	0.52	0.29	0.19	0.56	0.27	0.19	0.63	0.26
Avg.	**0.24**	**0.46**	**0.31**	**0.23**	**0.56**	**0.31**	**0.22**	**0.59**	**0.3**	**0.22**	**0.68**	**0.29**

Table 6.8 Average Precision, Recall and *F-Score* for CF-based methods using cosine similarity methods.

Average –PARAFAC1-EM												
	Top 3 Recommendation			Top 5 Recommendation			Top 10 Recommendation			Top 15 Recommendation		
	Pr	Re	Fs	Pr	Re	Fs	Pr	Re	Fs	Pr	Re	Fs
CL=10	0.34	0.61	0.43	0.31	0.56	0.39	0.14	0.66	0.23	0.13	0.68	0.22
CL=20	0.38	0.65	0.46	0.37	0.66	0.46	0.29	0.75	0.37	0.27	0.76	0.34
CL=30	0.32	0.55	0.39	0.29	0.56	0.36	0.24	0.61	0.31	0.22	0.64	0.29
Avg.	**0.35**	**0.6**	**0.43**	**0.32**	**0.59**	**0.4**	**0.22**	**0.67**	**0.3**	**0.21**	**0.69**	**0.28**

Table 6.9 Average Precision, Recall and *F-Score* for PARAFAC-1 using EM clustering.

Average –PARAFAC2-EM												
	Top 3 Recommendation			Top 5 Recommendation			Top 10 Recommendation			Top 15 Recommendation		
	Pr	Re	Fs	Pr	Re	Fs	Pr	Re	Fs	Pr	Re	Fs
CL=10	0.27	0.38	0.24	0.25	0.4	0.22	0.17	0.51	0.14	0.15	0.55	0.12
CL=20	0.23	0.45	0.3	0.22	0.48	0.3	0.12	0.71	0.21	0.11	0.72	0.19
CL=30	0.26	0.3	0.22	0.25	0.31	0.22	0.21	0.43	0.2	0.21	0.57	0.24
Avg.	**0.25**	**0.38**	**0.25**	**0.24**	**0.4**	**0.25**	**0.17**	**0.55**	**0.18**	**0.16**	**0.61**	**0.18**

Table 6.10 Average Precision, Recall and *F-Score* for PARAFAC-2 using EM clustering.

Average –PARAFAC3-EM												
	Top 3 Recommendation			Top 5 Recommendation			Top 10 Recommendation			Top 15 Recommendation		
	Pr	Re	Fs	Pr	Re	Fs	Pr	Re	Fs	Pr	Re	Fs
CL=10	0.36	0.55	0.42	0.32	0.57	0.38	0.29	0.69	0.33	0.27	0.7	0.31
CL=20	0.31	0.54	0.38	0.25	0.54	0.32	0.2	0.68	0.24	0.2	0.75	0.27
CL=30	0.34	0.58	0.41	0.3	0.61	0.37	0.23	0.7	0.3	0.21	0.71	0.27
Avg.	**0.34**	**0.56**	**0.4**	**0.29**	**0.57**	**0.36**	**0.24**	**0.69**	**0.29**	**0.23**	**0.72**	**0.28**

Table 6.11 Average Precision, Recall and *F-Score* for PARAFAC-3 using EM clustering.

Average –PARAFAC1-KM												
	Top 3 Recommendation			Top 5 Recommendation			Top 10 Recommendation			Top 15 Recommendation		
	Pr	Re	Fs	Pr	Re	Fs	Pr	Re	Fs	Pr	Re	Fs
CL=10	0.15	0.25	0.17	0.11	0.26	0.15	0.13	0.53	0.21	0.11	0.57	0.17
CL=20	0.27	0.39	0.29	0.3	0.39	0.31	0.24	0.67	0.31	0.22	0.7	0.28
CL=30	0.33	0.41	0.32	0.32	0.41	0.31	0.3	0.68	0.34	0.27	0.71	0.32
Avg.	**0.25**	**0.35**	**0.26**	**0.24**	**0.35**	**0.26**	**0.22**	**0.63**	**0.29**	**0.2**	**0.66**	**0.26**

Table 6.12 Average Precision, Recall and *F-Score* for PARAFAC-1 using KM clustering.

Average –PARAFAC2-KM												
	Top 3 Recommendation			Top 5 Recommendation			Top 10 Recommendation			Top 15 Recommendation		
	Pr	Re	Fs	Pr	Re	Fs	Pr	Re	Fs	Pr	Re	Fs
CL=10	0.19	0.41	0.26	0.25	0.41	0.26	0.13	0.66	0.21	0.11	0.74	0.18
CL=20	0.26	0.54	0.34	0.2	0.57	0.29	0.14	0.63	0.23	0.12	0.69	0.19
CL=30	0.20	0.45	0.27	0.17	0.54	0.25	0.15	0.65	0.23	0.13	0.75	0.21
Avg.	**0.22**	**0.47**	**0.29**	**0.21**	**0.51**	**0.27**	**0.14**	**0.65**	**0.22**	**0.12**	**0.73**	**0.19**

Table 6.13 Average Precision, Recall and *F-Score* for PARAFAC-2 using KM clustering.

Average –PARAFAC3-KM												
	Top 3 Recommendation			Top 5 Recommendation			Top 10 Recommendation			Top 15 Recommendation		
	Pr	Re	Fs	Pr	Re	Fs	Pr	Re	Fs	Pr	Re	Fs
CL=10	0.16	0.29	0.2	0.16	0.29	0.19	0.14	0.56	0.22	0.11	0.6	0.18
CL=20	0.24	0.46	0.3	0.2	0.55	0.28	0.18	0.65	0.26	0.15	0.7	0.22
CL=30	0.26	0.56	0.34	0.21	0.6	0.29	0.17	0.67	0.24	0.15	0.7	0.22
Avg.	**0.22**	**0.44**	**0.28**	**0.19**	**0.48**	**0.25**	**0.16**	**0.63**	**0.24**	**0.14**	**0.67**	**0.21**

Table 6.14 Average Precision, Recall and *F-Score* for PARAFAC-3 using KM clustering.

6.2.4 Discussion

Empirical results on '*Mileage Cars'* datasets show that the recommendation for all users suggested by the proposed tensor-based recommendation method outperforms the recommendations given by the traditional CF and hybrid methods, which mostly employ vector/matrix methods to find similarity between users. Taking the average of recommendations given by two-dimensional and tensor-based methods, on average there was an improvement of about 40% in the *Precision*, 52.78% in *Recall* and 42.64% *in F-Score* values.

From the results of group recommendations, it can be seen that recommendations based on two-dimensional methods based on Euclidian distance similarity measure perform worst in all the experiments. Due to the very high number of dimensions of interest vectors of users (742 dimensions excluding users) clustering based on Euclidean (ESM) using K-means and EM was unable to produce good clustering of users which ultimately resulted in only fair quality of recommendations. This can happen because as the number of dimensions grows significantly, ESM (Euclidian Similarity Measure) and CSM (Cosine Similarity Measure) start behaving differently and eventually become less similar. In very high-dimensional spaces as dimension gets higher ($\geq$ 128) [140], the two similarity measures start having small variations between them. However, the rate of decrease of similarity is very slow. Similarity vectors of instances in such high-dimensional spaces starts losing inter-component relationships, when traditional two-dimensional distance-based methods are employed. In comparison, cosine (CSM) measure produced average quality results. This happened because cosine similarity is able to map the different dimensions, but due to the two-dimensional model, some latent relationships between users-items were lost.

In contrast, the tensor-based methods are able to extract hidden relationships between the datasets and give much improved similarity results for the users-items data. For making CF-based recommendations, the traditional K-means algorithm with Euclidian similarity measure performed worse, whereas CSM methods performed at an average standard. Most notably, EM-based methods performed exceptionally well. These contrasting results confirm that distance-based approaches using Euclidian, Manhattan or cosine-based similarity measures used in high-dimensional data mappings reflect strange properties [165] which include loss of inter-component relationships and an inability to map inter-component latent relationships. EM is density-based clustering algorithm, unlike a distance-based clustering measure, it assigns a probability distribution to each instance, indicating the probability of it belonging to each of the other clusters.

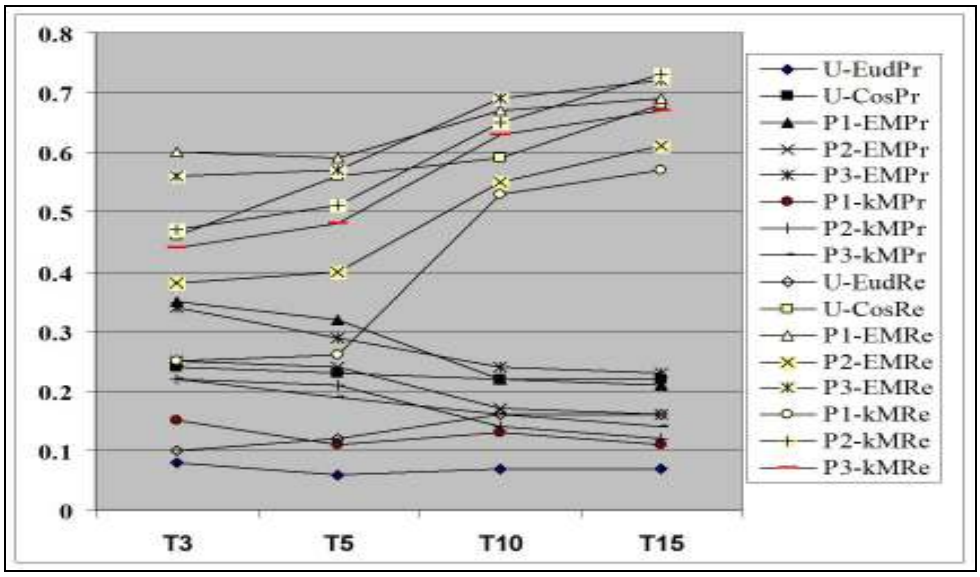

Figure 6.7 Precision versus Recall curve for top 3-top 15 (T3-T15) recommendations. (where suffix Pr=Precision and Re=Recall).

From the results shown in Table 6.6 and Figure 6.7, it is clear that, the number of best recommendations made to a user is around 3-5. As the number of *top* ϒ recommendations increases beyond the top 5, the value of *Precision* starts to fall gradually on the other hand as number of recommendation increase, the value of *Recall* starts to improve gradually. Thus, there is a choice between giving more recommendations and giving more accurate recommendations, which may be liked by a user. Based on all experiments and evaluation done in this research, it was found that the ideal number of recommendations to be made to a user would be around five, as both *Precision* and *Recall* curves have a maximum variation after this number of recommendations (Figure 6.7). The overall average *Precision*, *Recall* and *F-Score* for each datasets using simple CF-based methods, PARAFAC–EM and PARAFAC-kM based methods are shown below in Table 6.15.

Methods	Precision	Recall	F-Score
CF-Distance-based	**0.15**	**0.36**	**0.21**
PARAFAC-KM	0.16	0.52	0.24
PARAFAC-EM	0.26	0.59	0.36
Avg. PARAFAC-EM+KM	**0.21**	**0.55**	**0.30**

Table 6.15 Average aggregate recommendations for various methods.

The other noteworthy observation was that in most associations with large lengths (> 2), there was a reduction in the number of frequent item sets discovered in the process, which had high support and confidence values. Association with length=1 had too many rules, and such rules were biased towards rules with the highest frequency. Hence such rules were ignored for analysis in the experiments. An interesting observation (Figure 6.9) shows the relevant *F-Score* when the tensor best rank decomposition is considered. In the Figure 6.9, 1-EM, 2-EM, 3-EM refer to PARAFAC decomposition with best rank approximation of 1, 2 and 3 respectively, where clustering was achieved using EM clustering algorithm. Similarly 1-KM, 2-KM, 3-KM refer to PARAFAC decomposition with clustering achieved using K-means clustering method. In both cases, using EM and KM, performance starts decreasing with the increase in rank and number of recommendations.

In the case of KM clustering, rank 1 performs worst. KM clustering methods use distance measure for clustering. Due to the availability of singular values per instance for clustering, some useful relationships between instances may not be clearly distinguishable. When decomposing with higher ranks the extra factors available for clustering, have the ability to preserve some information by increasing separable distance between instances. Therefore the performance of KM clustering with higher ranks starts to increase. However, decomposition at higher ranks may start loosing valuable inter-component relationships, due to complete flattening of the tensor, and performance of TSM models may be similar or worse than the VSM methods. Therefore, tensor approximation up to rank three using two different clustering techniques, the K-means and the EM, so that one adopts a distance measure and other a density-based approach to cluster similar users are adopted in this thesis.

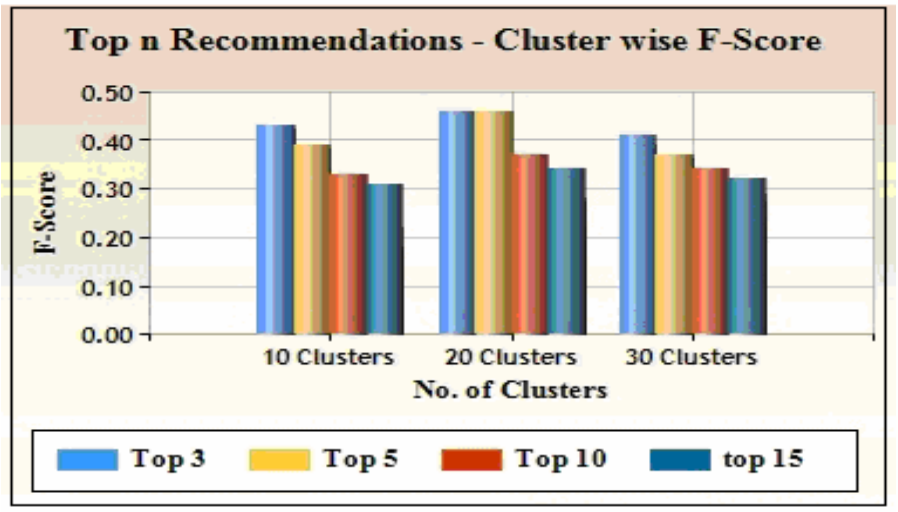

Figure 6.8 *F-Score* for the *top* ϒ recommendations cluster size wise.

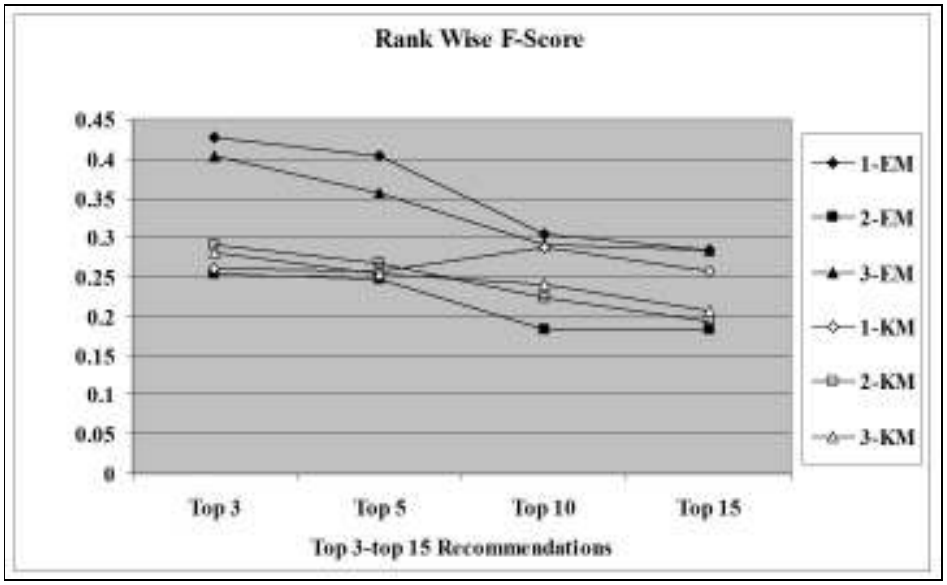

Figure 6.9 *F-Score* rank wise, *top* ϒ recommendations.

The Figures 6.8 & 6.9 clearly separates top 3 & top 5 recommendations from top 10 & top 15 recommendations. Rank 1 approximation using EM clustering performed the best among all cases. In the case of KM, rank 1 gave the best results when number of clusters was 30 and for rank 3, KM gave the best results when it was 30. These contrasting results in the case of KM are a clear indicator that distance measures may not work well when intra-distance between instances is small. Similarly when EM clustering is considered (Table 6.6), low top ϒ (3 & 5) performs best with low rank tensor decomposition (1-EM) whereas high top ϒ perform best with high ranks (3-EM).

Another observation is that distance measures need larger cluster sizes to maximize the distance between instances and hence improve overall performance. Experimental results clearly show that tensor-based recommendation methods and unique association rule generation for making recommendations outperforms the traditional CF-based methods, which evaluate user-items similarity measure using vector or matrix-based methods.

6.3 Comparison of Group and Individual Profile based Recommendations

Dataset 5 (Section 4, Table 4.5) is used to build both individual and group user profiles, where performance of recommendations made by individual profiles (Section 6.1) is compared with recommendations made using group user profiles. After various user models were created from this dataset, the twenty users, were grouped based on search behaviour and users within a group were given similar recommendations. Evaluation of individual personalized recommendations of each user is a cumbersome and lengthy process, especially when there are a large number of users. The search data belonging to these twenty users was taken for evaluation because of their typical Web search behaviour that represents most of the Web users, where some users make random unmatched searches, others make only a few searches and some others make a large number of searches.

6.3.1 Experimental Design

Individual user profiles for each user were created using tensors as discussed in Section 3.3.1. The *top* Υ items consisting of the make and model of a car were then given as recommendations to each user. Similarly, for group user profiles (Section 3.3.2), similar users were grouped and unique rules were ascertained for each specific group of users. The *top* Υ rules for each cluster were saved and these were given as recommendations to each member belonging to the same cluster.

6.3.2 Evaluation Methods

The evaluation methods for the individual user profile have already been discussed. (Section 6.1). To evaluate the quality of recommendations made to each of the individual user based on his group profile, experimentation was done with three different clusters of these twenty users. All association rules with high confidence values were taken for each cluster. Finally, unique rules were derived from these associations. Such unique rules were given as recommendations to each of the users in the group.

If subsequent searches made by each user matched the *top* ϒ group recommendations, the recommendations were considered accurate. The values of *Precision*, *Recall* and *F-Score* were evaluated based on the number of objects recommended and the number of objects matched. These values were evaluated in a similar manner as evaluated for an individual user profile. However, the only difference was that in this case, the recommendations that were made were based on similar users who were clustered in the same cluster.

6.3.3 Results

The results of group user profiles based recommendations results consisting of *Precison* and *Recall* values are shown in Tables 6.16-6.18. These results are shown when length of associations is two, and the top three objects in each cluster are taken as recommendations given to each user in the cluster. The results clearly outline that recommendations are more effective when using the Tucker models.

Rank	**Para1**		**Para2**		**Para3**		**Para4**		**Para5**		**Para6**	
	Pr.	**Rec**	**Pr**	**Rec**	**Pr**	**Rec**	**Pre**	**Rec**	**Pr**	**Rec**	**Pr**	**Rec**
Clust4	0.54	0.50	0.43	0.37	0.34	0.33	0.42	0.27	0.29	0.30	0.51	0.50
Clust6	0.36	0.31	0.42	0.45	0.27	0.51	0.27	0.34	0.39	0.48	0.42	0.48
Clust8	0.31	0.36	0.33	0.40	0.31	0.54	0.31	0.34	0.35	0.48	0.38	0.47
Average	**0.40**	**0.39**	**0.39**	**0.41**	**0.31**	**0.46**	**0.33**	**0.32**	**0.34**	**0.42**	**0.44**	**0.48**
	Average Precision for PARAFAC=0.37						**Average Recall for PARAFAC=0.41**					
F-Score=0.39												

Table 6.16 Precision and Recall values for GPF, using PARAFC rank 1-6, (where length of associations is 2).

Rank	Tuck1		Tuck 2		Tuck 3		Tuck 4		Tuck 5		Tuck 6	
	Pr.	Rec	Pr	Rec	Pr	Rec	Pre	Rec	Pr	Rec	Pr	Rec
Clust4	0.35	0.41	0.33	0.35	0.53	0.52	0.53	0.52	0.49	0.29	0.61	0.66
Clust6	0.36	0.46	0.30	0.36	0.44	0.50	0.38	0.33	0.43	0.48	0.48	0.48
Clust8	0.37	0.54	0.38	0.54	0.38	0.56	0.42	0.57	0.35	0.49	0.42	0.48
Average	**0.36**	**0.47**	**0.34**	**0.42**	**0.45**	**0.53**	**0.44**	**0.47**	**0.42**	**0.42**	**0.50**	**0.54**
	Average Precision for Tucker =0.42						**Average Recall for Tucker =0.48**					
F-Score=0.45												

Table 6.17. Precision and Recall values for GPF, using Tucker rank 1-6, (where length of associations is 2).

Rank	HOSVD1		HOSVD 2		HOSVD 3		HOSVD 4		HOSVD 5		HOSVD 6	
	Pr.	Rec	Pr	Rec	Pr	Rec	Pre	Rec	Pr	Rec	Pr	Rec
Clust4	0.43	0.41	0.26	0.29	0.36	0.47	0.36	0.47	0.27	0.50	0.27	0.50
Clust6	0.24	0.42	0.32	0.32	0.24	0.37	0.25	0.34	0.23	0.32	0.39	0.54
Clust8	0.23	0.50	0.20	0.25	0.22	0.38	0.19	0.36	0.19	0.36	0.30	0.42
Average	**0.30**	**0.44**	**0.26**	**0.29**	**0.27**	**0.41**	**0.27**	**0.39**	**0.23**	**0.39**	**0.32**	**0.49**
	Average Precision for HOSVD=0.28						**Average Recall for HOSVD =0.40**					
F-Score=0.33												

Table 6.18 Precision and Recall values for GPF, using HOSVD rank 1-6, (where length of associations is 2).

The group user profiles based recommendations consisting of *Precison* and *Recall* values are shown in Tables 6.19-6.21. These results are shown when the length of associations is three and the top three objects in each cluster are taken for recommending to each user belonging to the cluster.

Rank	Para1		Para2		Para3		Para4		Para5		Para6	
Rank	Pr.	Rec	Pr	Rec	Pr	Rec	Pre	Rec	Pr	Rec	Pr	Rec
Clust4	0.31	0.07	0.49	0.28	0.35	0.05	0.45	0.17	0.34	0.03	0.33	0.29
Clust6	0.27	0.15	0.22	0.17	0.28	0.33	0.29	0.19	0.22	0.02	0.25	0.19
Clust8	0.23	0.23	0.15	0.19	0.42	0.36	0.24	0.23	0.22	0.27	0.23	0.25
Average	**0.27**	**0.15**	**0.29**	**0.21**	**0.35**	**0.25**	**0.33**	**0.20**	**0.26**	**0.11**	**0.27**	**0.24**
	Average Precision for PARAFAC=0.30						**Average Recall for PARAFAC=0.19**					
F-Score=0.23												

Table 6.19 Precision and Recall values for GPF, using PARAFC rank 1-6, (where length of associations is 3).

	Tuck1		Tuck 2		Tuck 3		Tuck 4		Tuck 5		Tuck 6	
Rank	Pr.	Rec	Pr	Rec	Pr	Rec	Pre	Rec	Pr	Rec	Pr	Rec
Clust4	0.26	0.51	0.28	0.05	0.28	0.13	0.28	0.13	0.31	0.38	0.27	0.31
Clust6	0.15	0.17	0.18	0.18	0.25	0.2	0.25	0.21	0.26	0.18	0.29	0.07
Clust8	0.13	0.26	0.26	0.29	0.20	0.51	0.18	0.36	0.19	0.35	0.22	0.17
Average	**0.18**	**0.31**	**0.24**	**0.17**	**0.24**	**0.28**	**0.24**	**0.23**	**0.25**	**0.30**	**0.26**	**0.18**
	Average Precision for Tucker =0.24						**Average Recall for Tucker =0.25**					
F-Score=0.24												

Table 6.20 Precision and Recall values for GPF, using Tucker rank 1-6, (where length of associations is 3).

Rank	HOSVD1		HOSVD 2		HOSVD 3		HOSVD 4		HOSVD 5		HOSVD 6	
	Pr.	Rec	Pr	Rec	Pr	Rec	Pre	Rec	Pr	Rec	Pr	Rec
Clust4	0.32	0.24	0.2	0.05	0.38	0.4	0.38	0.4	0.22	0.38	0.22	0.38
Clust6	0.2	0.29	0.23	0.24	0.2	0.25	0.16	0.26	0.22	0.24	0.37	0.33
Clust8	0.18	0.33	0.03	0.09	0.16	0.3	0.17	0.3	0.17	0.3	0.27	0.34
Average	**0.23**	**0.29**	**0.15**	**0.13**	**0.25**	**0.32**	**0.24**	**0.32**	**0.20**	**0.31**	**0.29**	**0.35**
	Average Precision for HOSVD=0.23						**Average Recall for HOSVD =0.29**					
F-Score=0.26												

Table 6.21 Precision and Recall values for GPF, using HOSVD rank 1-6, (where length of associations is 3).

Table 6.22 shows the average results of recommendations of VSM and TSM based methods when group user profiles are considered. In the case of vector methods, when the length of associations is two, the performance of VSM methods is far inferior to the tensor methods. However, when the lengh of associations is three, all the methods have comparatively similar *F-Scores*. This is a clear indicator that deciding upon the length of associations to be chosen when making recommendations is essential.

Method	Length of Association=2	Length of Association=3
VSM	**0.24**	**0.23**
PARAFAC	**0.39**	**0.23**
Tucker	**0.45**	**0.24**
HOSVD	**0.33**	**0.26**

Table 6.22 Average *F-Score* for GPF based methods taking different association lengths.

In cases where individual user profiles are compared with group user profiles (Where length of associations is taken as two, and where tensors are taken up to rank three decomposition, for both individual user profiling and group user profiling), the results in Table 6.23 clearly show that individual user profiling methods gives recommendations which are more accurate than the recommendations given based on group profiles.

Method	PARAFAC	Tucker	HOSVD
TSM Individual Profiles.	0.48	0.48	0.47
TSM Group Profiles.	0.40	0.40	0.37
% Improvement	20%	20%	18.9%

Table 6.23 Average *F-Score* for individual user profile and group profile methods.

6.3.4 Discussion

When the same users search data was taken to build group user profiles, it can be clearly seen that individual user models were able to recommend better objects. In the case of group user models the best F-Score was 0.40, whereas in case of the individual user model, even the worst score was higher than this. However, when using a tensor-based group profile for recommending, the F-Score of the top three recommendations was far superior when compared to the average matrix methods. Good quality of clustering resulted in good recommendations made to users. This is clear from the result sets.

When tensor-based group profiles were compared with matrix-based individual profiles, clearly in most of the cases the performance of even group profiles was better than the individual profiles of vector-based methods. However, when the average F-Score of matrix-based methods of individual user profiles is taken (Refer to Table 6.4 top 3, which is 0.40), and the average score of group user profiles with a length of associations as two is taken (0.39), the marginal difference in the values of the two confirms the fact that the clustering achieved using tensor decomposed values was able to cluster users more cohesively, resulting in good clustering. In contrast as can be seen from Table 6.22, when matrix methods were used to build group profiles tensor-based group methods were far superior to these two-dimensional methods.

The other factor that greatly effects the quality of recommendations made to users is the number of associations to be considered when choosing rules. Associations with a large number of itemsets results in poor quality of recommendations as very few rules are extracted. In all the results shown in Table 6.22, the best recommendations were delivered to users when the number of rules was two, whereas when the length of associations was taken as three, the performance of recommendations started declining sharply no ,matter what techniques was used for making recommendations.

One major conclusion that is clearly visible in most of the results is that personal agents are capable of delivering better recommendations then agents based on the CF approach, no matter which method is used for recommendations.

6.4 Recommendation based on Object Profiles

Object profiles are groups of objects or items that are clustered together based on similarity of one or many features. Two objects as in the case of two cars may have the same colour, make and price. When clustering objects the best cluster of objects is obtained if all objects with maximum similarity among each other are clustered in the same group.

6 .4.1 Experimental Design

To evaluate the effectiveness of recommendations using object profiles this thesis has used '*Dataset 5*'. Details about datasets are discussed below.

Dataset 5: The purpose of using this dataset is to evaluate the performance of recommendations when object models are used in conjunction with a group user profile model. For all experiments using object models, the cars as searched by users belonging to *Dataset 5* are taken for constructing the model. All distinct cars, 429 in number as searched by the 20 users are taken to build the tensor model. The object structure based on which tensor was created is:

$$Object_{Tensor} = Kms \times Transmission \times Engine\ \text{size} \times Fuel\ \text{type} \times Location \times Car\ Id$$

The final values for each dimension for this dataset are shown in Equation (54).

$$\mathcal{T} = M^{8\times10\times9\times8\times6\times429} \tag{54}$$

Once the car model is created, it can be decomposed using PARAFAC, Tucker or HOSVD and the values obtained from the matrix $\mathbf{M}_n$ (as discussed in Section 3.1.4 and Figure 3.10) are then used to cluster similar cars. When making recommendation to users, cars belonging to a same cluster are recommended based on number of query features matching with the cars in the cluster. Highly matched cars are ranked higher when making recommendations.

6 .4.2 Evaluation methods

For comparing the object tensor of cars, the baseline used was traditional two-dimensional vector methods. All cars based on the filtered attributes such as mileage,

transmission, engine, fuel, and locations were clustered using K-means clustering algorithm. To evaluate the quality of top Υ recommendations given by each method the following metrics based on *Precision* and *Recall* values was used. Let S_z be the set of searches made by user u_z after the model is created, and let $\Re_z^{\Upsilon}$, be the set of *top* Υ recommendations given by various methods to u_z, where $\Upsilon \geq 3$ *and* $\Upsilon \leq 5, \Upsilon \in \{\{3\},\{5\}\}$. For evaluation purpose only the top 3 and top 5 recommendations for object profiles were taken, because the focus is on the quality of recommendation rather then quantity. Precision (Pr_n) and recall (Re_n) for each user u_z was evaluated as

$$Pr_z = \frac{\left|\Re_z^{\Upsilon} \cap S_z\right|}{\left|(\Re_z^{\Upsilon} \cap S_z) + (\Re_z^{\Upsilon} - S_z)\right|} \tag{55}$$

$$Re_z = \frac{\left|\Re_z^{\Upsilon} \cap S_z\right|}{\left|(\Re_z^{\Upsilon} \cap S_z) + (S_z - \Re_z^{\Upsilon})\right|} \tag{56}$$

6.4.3 Results

The following Figure 6.10 summarizes the results of recommendations given by various methods. The baseline is purely a two-dimensional data modelling approach, where objects have been clustered based on similarity of features. In the Table GPF means group user profile and OBF refers to object profile.

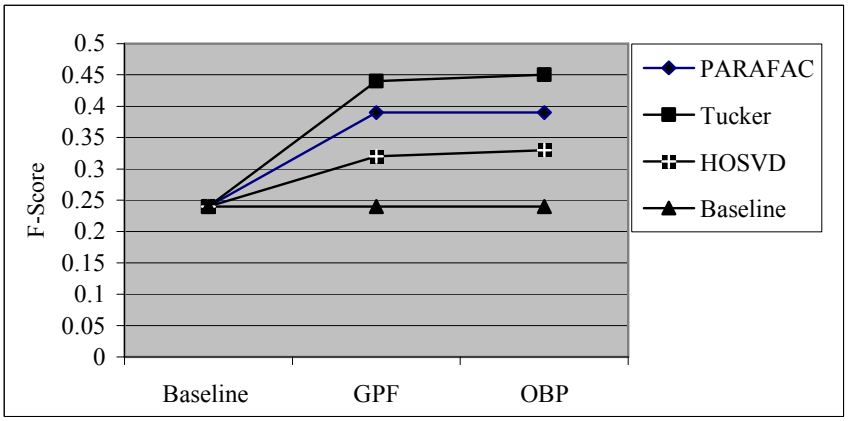

Figure 6.10 Overall average summary of recommendations results.

6.4.4 Discussion: Recommendation Methods

Results in Figure 6.10 clearly show that the quality of recommendations improves when the recommendation method utilizes the object profile information. However, the improvement was marginal. In the case of group user profiles, similar users were recommended cars based on their group profile, and in the case of the car object model, users were recommended similar objects from the object cluster. The experiments indicated that if most relevant and important dimensions are taken and suitable data modelling methods such as tensors are used for the object modelling, then object modelling can give better results than recommendations based on group user profiles. In terms of tensor modelling the recommendations made using Tucker methods preformed the best.

6.5 Summary: Recommendation Methods

When a user makes a search, the results given to him can be enhanced by giving him personalized recommendations matching his needs. The quality of recommendations being made to a user mostly depends on his user profile. Methods that are capable of highlighting a user's behaviour more prominently are needed to build user profiles. User profiles that can be fully mapped to a user's actual behaviour compromising of multiple dimensions are needed. Traditional two-dimensional methods used for finding the vital relationship between users and items and building user profiles are not adequate to map these relationships. Clearly as can be seen from most of the results MDD methods are far superior to two-dimensional methods for making recommendations. The key conclusions that are derived from observation of results of recommendations based on individual profile, group profile and object profiles are listed below.

1. Unlike previously used methods of recommendations based on two-dimensional models [73], [99], [152] these methods are not of high quality as compared to recommendations based on MDD methods.
2. Individual personalized recommendations are more effective then recommendations based on group or object profiles.

3.When group and object profile based recommendations are compared, object profile shows a slight improvement in the quality of recommendations. This happens because unlike group profiles which have similar objects as searched by like minded people, object profiles have similar objects as searched by a user.

4. Most of the previous works have used either user profile, group profile or object profile [10], [28], [36], [94], [136] however, this thesis proposes to utilize all available information. The three categories of profile the individual user profile, group users profile and the object profile can be used in conjunction to improve the quality of recommendations. The Individual profile helps in catering to the personnel needs of a user, the group profile can bring interestingness by recommending new and unique items and the object profile can be helpful especially in making recommendations to new visitors.

Once, *top* ϒ recommendations are selected from a user profile, the next most important objective of any Web personalized system is to adopt a methodology to rank these recommendations and present them in an ordered manner. Section 2.4.2 of the thesis detailed about some currently adopted methods of ranking Web search results. Recommendations have to be ranked based on the user's current search needs while keeping the user's preferences (derived from his various profiles) in consideration. If the user is a new visitor to the website, then search results can be ranked based on his current search query.

This chapter discusses the experiments conducted to measure the effectiveness of the ranking methods. To achieve this, two types of evaluation experiments were conducted. The first set of experiments tested the the effectiveness of the proposed FIT (Feature Importance Technique) in comparison to other widely adopted Web/database ranking methodologies. The other set of experiments tested effectiveness of personalized search over the normal Web database search.

Section 7.1 of the chapter discusses about how the proposed FIT method was evaluated in a parameterized query environment. Section 7.2 evaluates the performance of the FIT in a personalized search environment.

7.1 Evaluation of Results on Parameterized Web Search

In a parameterized Web search environment, when a user directs a search query, the results that are returned are given based on matching the SQL query to the results that are present in the database. A big problem arises when too many results or no results are returned for a user's query. In these situations it becomes very difficult to filter the information and provide most relevant information that is closest to a user's search.

In this evaluation method a qualitative study was undertaken to check the effectiveness of rankings suggested by FIT by comparing with the SQL database engine (or database search with order by clause in descending order of preferences given by a user) and other different ranking methods such as FIT, IDF (Inverse Document Frequency) [8] , QF (Query Frequency) [8] and QC (Query Classification) [187]. IDF extends TF-IDF concept of an IR model to rank database tuples. Database

tuples are compared with query attributes, after that a distance-based metric such as Euclidian or cosine is used to calculate the similarity of tuples to a user's query.

In the QF method the importance of attribute value is determined by the frequency of its occurrence in the workload. Workload consists of all searches made by users that are stored in the database. On the other hand QC will retrieve results from other similar queries classified in the same category and which are closest to the given query by the user. For methods like FIT, IDF, QF and QC the *top* Υ results were taken in descending order of similarity score.

7.1.1 Experimental Design: Parameterized Web Search Ranking

The Table containing car information was 3.8 GB in size and had about 42 million rows and 63 attributes or features containing information about cars.

Seven users were involved in the testing phase of experiments. Each user was asked to submit five queries to the main cars database, containing 1,2,3,4 and 5 parameters. For each query with more than one searched feature, users were asked to give the order of preferences for individual features from high to low. The top twenty records for each query were retrieved using different database ranking methods like SQL, FIT, IDF, QF and QC. The top 20 results given by each method were given to the users for evaluation without the details of the method. Users were asked to rate these top twenty based on the NDCG (Normalized Discounted Cumulative Gain) scheme [6], where a score of 4 meant perfect, 3 meant excellent, 2 meant good, 1 meant fair and 0 meant bad.

The feature cost was used as the query classification (QC) criteria, where each query was classified according to the cost of the car e.g. Low = $1-25000, Medium = $25001-50000, High = $50001-100000 and Premium = above $100000. A total of seven different test query sets consisting of five queries per user as given by each user were used for evaluation. The queries were classified into the four categories as shown in Table 7.1.

Query Classification (QC)	Searched Parameters	Operators used
Low Cost=30%; Medium Cost =35% ; High Cost=25%; Premium Cost=10%;	Make, Series, Cost, Body type, Engine size , Drive	=,<,>, ≤,≥, And, Between, Like, Or, %%, IN

Table 7.1 Query selection criteria

All search queries consisted of at least a single feature or a combination of features such as Make, Series, Cost, Body type, Engine size, Drive. The remaining features (a total of 57 attributes) were other attributes such as seating, transmission, zip etc. These features were not retrieved in the query results and thus were not evaluated by the seven users when rating the accuracy of results given by each query.

7.1.2 Evaluation: Parameterized Web Search Ranking

This thesis has used NDCG to measure the effectiveness of various ranking methods. NDCG is specifically suited for Web search evaluation. Let Υ be the top ranked results. The ranked results for a given query Q are scored from top to bottom. NDCG for a given query Q is evaluated as shown in Equation (57).

$$N_Q = M_Q \sum_{j=1}^{\Upsilon} (2^{r(j)} - 1) / \log(1 + j) \tag{57}$$

Where M_Q is a normalization constant whose value is 1 for a perfect ordering and each $r(j)$ is value of rating (Eg. 0=Bad, 5=Perfect) at position j. Previous research have shown that NDCG ranks highly rated results higher and is thus well suited to Web search evaluation [6].

7.1.3 Results and Discussion: Parameterized Web Search Ranking

Judging by the results in Figure 7.1 the FIT based ranking method performs consistently well in nearly all the cases, when a different number of parameters or features are searched. This happens because there is a complete utilization of the workload and the preferences given by the users. Unlike other methods, which use either workload or only user preferences, the FIT, takes into consideration all inputs from the user. Thus, it gives the best rankings. In situations where the user preferences are not specified or no profile information is available, FIT still manages to achieve the best results comparative to all methods discussed. In these cases, it matches all attributes and ranks results using the CF approach utilizing the workload information. In cases where a single attribute was searched by users all methods gave nearly comparable results. However the ordering of result sets achieved by different methods was different.

When similarity for numeric attributes is being evaluated, as proposed by [8], IDF may not be very effective, as evaluating numerical similarity measures are not well defined. Apart from this limitation, IDF is biased towards documents with higher occurrences. One of the reasons for its mediocre performance may be the lack of a high number of similar occurrences in workload as searched by the users. The performance of IDF based ranking is nearly average in the overall performances of all metrics. IDF works better than QC methods in nearly all the cases.

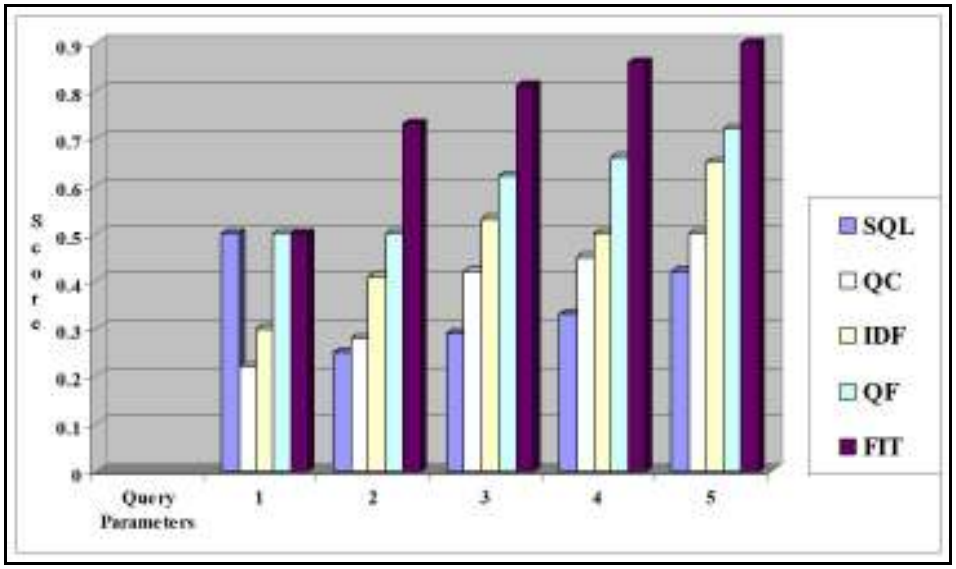

Figure 7.1 NDCG Comparative results of various ranking methods.

QF is purely workload dependant and has various disadvantages especially in cases when workload information is unreliable [8]. QF based ranking may work well in most of the scenarios however, in cases when a user's query is unrelated to his previous workload, then QF will fail to retrieve good rankings. For example, if a user had searched for sedan cars in the past, all workload based ranking methods would rank such cars higher, even though the user may have currently searched for an entirely new body type like a 'utility' vehicle.

QC performed worst in the absence of clear classification parameters (such as no cost criteria was given by users in their search). Overall QC performed slightly better than a normal SQL search. This happened because QC retrieved results from other similar queries classified in the same category and which were closest to the input query by the user, whereas SQL fails to retrieve results when the query is rigid.

As can be seen in Figure 7.1, FIT top ranked result sets perform the best in all cases. In case of categorical parameters when the number of the searched parameter is more than one, all FIT with score > guaranteeing at least one match. For numeric attributes, FIT with score > 0 would match such records based on a decreasing order of similarity to the values as searched by users. As the number of searched parameters increases, the range of records displayed to a user increases significantly, because even a single match of attribute obtains some score, and therefore the number of results retrieved increases. FIT will always retrieve rows as long as there are some matching attribute values to a user's query in the database. Thus, it can handle *empty-answers* problem effectively. In all cases, the *top* ranked results given by FIT, guarantee the best match to a user's query. The *top* ϒ results ranked by FIT perform exceptionally well.

In the case of purely categorical attributes being selected, when the number of features searched is one, all such records, which match perfectly to the searched attribute, would be ranked higher. In the case of two or more attributes being selected, the top rank would always contain the highly-matched result sets. When two attributes are selected, it would contain at least one attribute fully matched. Thus in the case of six attributes being selected it would always contain five, four or three depending upon the number of features matched. Apart from this, it would score the results based on importance of features as derived from workload.

Even in the worst scenario for all categorical attributes it will have at least one feature matched to a user's query, if it exists in the database. Therefore, unlike the previous methods such as QFIDF [8] if no match of attributes occurs, then no score is given. Thus, such unmatched attributes do not play any role and are ranked lower in the overall search results displayed to a user.

In case of ranking purely numerical attributes, ranking is based on percentage similarity between searched and queried features. Perfect similarity between attributes obtains a score of 1, and a score of 0 means either one of the features is 0 or null. If there is insufficient workload information FIT has the ability to score based on the current search preferences as specified by a user. This scoring can be provided explicitly in the procedure implementing the ranking. Good Ranking of results as obtained from the test experiments, in all cases where users may have correctly classified their preferences for a query, are a clear indicator of the success of this approach.

7.2 Evaluation of Personalized Web Search

Personalized Web search is becoming popular, where a user's past preferences are kept in consideration when displaying or recommending results to a user. In personalized Web search a user's previous search interests are saved in his user profile and this information then helps in mapping his interests with his searches.

7 .2.1 Experimental Design: Ranking Personalized Search

In the first evaluation method a comparison between results given by a personalized search and normal SQL search, which is the most common search adopted by many websites is discussed. In this evaluation method, user profile information is used to give search results to a user.

To evaluate the effectiveness of ranking based on user profiles, all user profiles built from Dataset 5 were considered. The user profile of each user had information about searched parameters like make, model, body type, search type and cost and what were the user's *top* ϒ preferences for each of these search parameters. A set of three search queries with highest TF-IDF were taken from the workload for each user for experimentation and evaluation. The higher a particular parameter was searched, the higher importance it gets.

7.2.2 Evaluation: Personalized Search

To evaluate the quality of ranked recommendations, personalized search recommendations were compared with a user's actual searches. For each user all actual searches were taken only after the user profiles were created. To calculate the similarity between a user's actual search and personalized search the various searched components were compared. Two of his searches were considered similar if each had same car make, model, body type, search type and cost. These user searches were then expanded as per his individual profile (Section 3.3.1). The profiles contained information about various objects preferred by a user for example, whether the user preferred a sedan car to a utility or a four-wheel drive (Table 7.1). Each parameter, as searched by the user was given a level of importance based on his profile information. When making recommendation to each user the individual preferences listed as *top* ϒ

in his profile were considered. As an example when making ranked recommendations to the user u_1 (identified as user id=1 in Table 7.2), all Ford sedan's with cost range of C1 would be rated higher.

User Id	Make		Body type		Cost Range	
	Name	Score	Name	Score	Name	Score
1	Ford	1	Sedan	1	C1	1
1	GM	0.75	SUV	0.75	C2	0.78
1	TA	0.56	Hatch	0.56	C3	0.57
2	Ford	1	Sedan	1.00	C4	1.00
2	GM	0.78	Hatch	0.56	C5	0.80

Table 7.2 A sample profile of two users.

From the server workload, the searches of the same twenty users for the next three days excluding the profile creation data were taken as a benchmark. On average, there were eight searches per user for benchmarking. For each of the three test queries of each user, the top ten rows with the highest scores were retrieved by the FIT algorithm and were given as search results to the users. Apart from this the normal search given by a SQL database query, for the entire three test queries of each user were also used as a baseline measure. These results given by the FIT and baseline were compared with the benchmark. *Precision* and *Recall* was based on these, and their values were identified as:

False Positive (F.P): Results returned by FIT (Highest Score) or baseline and not matching even a single search of a user from the list of results in the benchmark.

False Negative (F.N): Benchmark had records of a user but neither FIT (Highest Score) nor baseline retrieved the same results as in the benchmark.

True Positive (T.P): Number of searches given by FIT (Highest Score) or baseline matching the benchmark search results. Thus PR was calculated as

$$\Pr ecision = \frac{T.P.}{T.P. + F.P.}. \ , \ \mathrm{Re}call = \frac{T.P.}{T.P. + F.N.}.$$

7.2.3 Results and Discussion: Personalized Search

The results in Table 7.3 clearly indicate that a personalized search improves the overall quality of search results given to a user. Another observation which was evident from the experiments was that each search parameter as searched by a user

plays a different role. Some searched parameters are more important and play an important role in distinguishing between the preferences of two users. This can make a difference when displaying results to the users. Two users may search using the same search criteria but the results that can be displayed to them can be differently ranked based on the personnel preferences of each.

No. of Users	Average Precision Baseline	Average Precision FIT	Average Recall Baseline	Average Recall FIT	Average F-Score Baseline	Average F-Score FIT
20	0.24	0.36	0.40	0.43	0.30	0.39

Table 7.3 Results of personalized search with baseline search on Dataset 5.

The higher value of *Precision* in personalized searches clearly indicates that the given results are more likely to be searched by users, as it matches to some extent their search behaviour. Overall, the low *Precision* value in both the cases is because there were not as many searches for benchmarking as there were ranked recommendations given to them by the FIT and baseline. Due to Web users unpredictable search behaviour, the value of *Recall* is low in both the cases. A user may enquire about certain objects and then may not search any of the retrieved objects. A very similar *Recall* value in both the cases confirms this.

However, in personalized searches there is a marginal increase in the value of *Recall*, which indicates that personalized searches may give results which users are more likely to search. All of the ranking experiments showed an improvement of about 23% in terms of *F-Score* over the standard Web-parameterized search when FIT method was used in conjunction with user profile information for retrieving personalized results.

7.3 Summary: Ranking Methods

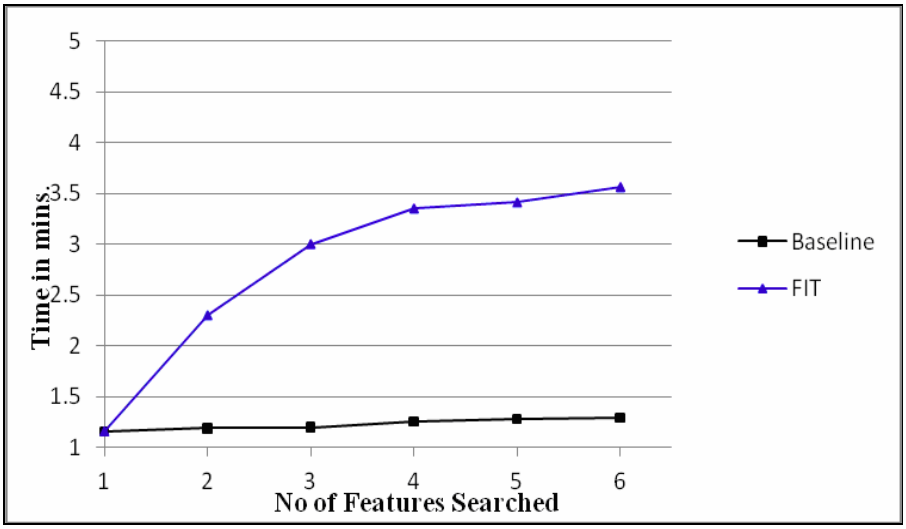

Figure 7.2 Time Comparisons between FIT and Baseline.

The FIT algorithm has a time complexity of *O(n),* where n is the number of parameters searched. When the number of parameters searched is one, FIT takes nearly the same time as the SQL engine. For $n > 2$, the time taken to execute the FIT algorithm increases linearly as the number of parameters searched is increased (Figure 6.2). On an average, the FIT took around 2.5 times more time for each query than the normal database query. Thus, there is a trade-off between the quality of ranked results returned and the time spent to get such results. However indexing coupled with fast processing speeds and improvements in hardware technology can significantly improve the execution time of the FIT ranking algorithm.

Whether it is a plain database search, Web search or personalized Web search the ranking of result sets is crucial to give the best results as needed by a user. All results which have higher relevance for a user have to be ranked higher so that the user has easy access to such information. This thesis proposes FIT as a ranking function that has the capability to give ranked search results. The proposed method can be used to give ranked search results based on relevance to a user's needs. The FIT algorithm is flexible and keeps a user's current search at the highest level of preference. In the case where the user has profile information, the proposed method gives personalized search results based on a user's current needs and his past preferences. The key findings derived from ranking experiments are listed below.

1. Ranking methods are a key component of the whole personalized system. Most of existing systems just give plain recommendations without ranking the results. Apart from this, unlike the proposed FIT ranking method, most lack the flexibility. FIT can be used for ranking recommendations based on a user's profile or providing *top* Υ results to a user just based on his search query.
2. Ranking methods can be enhanced by using user profile information. When ranking recommendations, user profile information can be used to rank highly preferred items higher.
3. In case of users who do not have a profile, ranking methods can be helpful in ranking results based on user's current query. All searched parameters can be ranked higher and the corresponding results can be ordered in a ranked manner.

Ranking is an essential component of any Web based personalized system, however, most of the existing Web personalized systems (WPS) do not have a detailed methodology to handle this [42], [44], [93], [195]. Seldom are ranking methods used by these systems to order best results as which may be highly liked by a user. However, this thesis proposes a ranking metyhodology that can enhance the recommendation capabilities of a WPS.

Chapter 8 Conclusion and Future Work

Web servers at different locations around the world receive huge traffic. Due to this enormous traffic volume and chaotic access behaviour of Web users, it is very difficult and complex to predict the access patterns of users. When choosing information online, consumers always have a choice and this is very true in regard to information distribution on the Web. The same information can be provided by many Web service providers in different order and it is up to the user to choose the one which best suits his/her needs. Gaining and retaining Web site visitors is a major concern for every Web service provider. Personalization of services focussing on individual user needs is a key to gain and retain visitors.

The ultimate aim of personalization is that the complete personalized system should be able to generate effective recommendations for the individual user or group of users. These recommendations should have a logical reasoning that matches with the user's profile and the search object. These profiles should consider the multi-dimensionality of users search behaviour and search objects. The methods used to build profiles should be able to outline the important relationships between users-users and items-items. Thus, recommending similar users near similar items in a ranked order, based on the personal preferences of each user, and keeping current search interests of a user at the highest level are some of the major objectives of any Web personalization system.

The major objective of the thesis was to compare widely adopted two-dimensional data modelling and profile building strategies with the multi-dimensional data modelling and profile building techniques. Since, Web server log data is MDD in nature, utilizing two-dimensional profile building methods on this data may result in poor quality of user profiles. Therefore, to improve profile building methods, this thesis proposed MDD techniques based on tensors to build user profiles.

The other objective of the thesis was to utilize various profiles such as individual user, group user and object profiles to make recommendations to users. Most of the existing recommender systems do not use any ranking methodology to return an ordered list of searched items to users. Ranking the recommendations as per a user's preferences was another objective of this thesis.

8.1 Contribution

More specifically, the following are the contributions of this research thesis.

1. Web log data is complex multi-dimensional data, consisting of one or more searches of many users with multiple searched features and each feature having multiple values. Most existing methods use two-dimensional methods to model and mine knowledge from Web log data [128]. Two-dimensional methods are unable to map such high-dimensional data due to the lack of availability of appropriate representation methodology and inability of two-dimensional methods to interact with various dimensions freely. Thus two-dimensional methods are unable to retain latent relationships that exist between various dimensions [165]. MDD models using more than three dimensions to model and mine patterns from Web log data and build user profiles, have rarely been used. Users and items/objects are core constituents of a user profile. Finding closest similarity between each is very important. To find the closest similarity between users-users, user-items and items-items this thesis proposed MDD techniques like tensors for modelling user and object behaviour.

2. The other contribution outlined in this thesis was introducing methods of building various users, group and object profiles. Proposals for building each kind of profile creation method starting from data pre-processing; relevant feature identification based on log data and website analysis; feature extraction; modelling the data into an MDD model; and subsequently building profiles have been outlined while keeping the MDD nature of Web log data under consideration.

3. Clustering of similar users and objects is important for Web personalization systems. For clustering similar users, this thesis proposed to do clustering on the last dimensional matrix using the proposed DIF (Dimension Influencing Factors) method. In the case of objects with multiple attributes and with mixed data types (numeric, categorical or both), to improve the quality of clustering solutions obtained, the thesis proposed to cluster such objects using the FIBCLUS similarity measure, so that inter-object distances are maximised and intra-object distances are minimized, thus enabling good clustering solutions.

4. Unlike most of the personalized systems, those use either the user model, or the CF approach, or both, this research proposed to use multiple profiles (individual, group and object profiles) in conjunction to make best possible recommendations to different categories of users.

5. Ranking methodology adopted by personalized Web recommender systems is crucial to obtain the best results for matching a user profile. Most of the existing Web personalized systems do not fully use all available information to extend the capability of such systems. This thesis proposed a ranking methodology that can recommend based on user's current search; use individual profile information to enhance the search; and in case no profile information is available; it can use the CF approach to give ordering of results matching a user's need.

8.2 Research Findings

Some important findings of this research are

1. Two-dimensional data modelling techniques may not be good enough to model Web users log data (due to its multi-dimensionality), when used for modelling user behaviour. When recommendations were made using both individual user profiles and group user profiles in our experiments, tensor-based models performed far superior to the two-dimensional methods in both cases. This research work has done extensive research on data modelling in MDD environments, having more than three dimensions and have found out that these models can be highly effective compared to existing two-dimensional models used for mining information from MDD. When an average *F-Score* of top 3, 5, 10 and 15 recommendations for both vector and tensor-based methods were compared, when tensor based methods were used for making recommendations there was an improvement of about 31.74% in the quality of recommendations made to individual users. When tensors were used instead of vector and matrix-based methods for group profiles, there was an improvement of about 36.84% in the average *F-Score*.

2. Traditional clustering on MDD may not be appropriate for obtaining the best clustering solutions, as most of these methods cluster vectors of user's interests, represented by their searches. Due to the multi-dimensional nature of Web log data the relationships exposed in terms of inter and intra-cluster similarity may not be fully

exploited when clustering is employed on these datasets. On the other hand when these datasets are first modelled keeping their MDD nature in consideration and then dimensionality reduction is done to highlight the inter-object relationships, the clustering solution obtain therein is much better than employing traditional clustering technique. Therefore, such data need to be modelled using some MDD modelling technique. In all our experiments clustering results obtained on various datasets, confirm this observation. Furthermore, in most of the cases the proposed DIF method gives clustering solutions that are better than vector and tensor based methods. DIF has the ability to improve clustering because it adds an extra distance component to be considered when clustering the search data of users. In the case of clustering objects, it was found that the proposed FIBCLUS similarity measure enhances clustering by reducing the inter-cluster similarity distance and increasing the intra-cluster similarity distance between object instances. Results on various test datasets further proved this.

3. When finding interesting patterns of individual user and group users searches, sequential rule and association rule may not be good enough to be stored as *top* Υ recommendations. Sequential rules may give rules, which are rare within a group, and association rule may give rules, which are redundant. Therefore, unique rules with the high confidence values have to be extracted and stored in the various user profiles. The other finding is that for associations of greater length (two in our case), performance of recommendation starts to decline. Therefore, when finding associations rules, the length of rules is an important factor and can play a crucial role in the overall performance of a recommender system.

4. Recommendation using individual profiles is superior to that of using group profiles, and in the case where profiles are used in conjunction, the performance of recommendation improves. Individual profiles reflect a user's personnel preferences whereas group profile reflects a group's interests. The quality of recommendations will definitely improve when an individual user is given personnel attention. In case when profiles are used in conjunction, the performance of recommendation improves. Since, group profiles reflect behaviour of similar users in a group, group profiles can be used to recommend unique items that may interest a user. On the other hand when object profiles are used in combination with individual or group user profiles, the users are given near similar objects as which may interest them. When similar objects as searched by a user and matching his personnel preferences are given as

recommendations, it is more likely that a user may search these objects than search unrelated and unmatched objects.

5. Deciding on *top* Υ recommendations and how such recommendations can be made is very important. From the analysis of users profile (individual and group user profiles), in this thesis, it was deduced that the best number of recommendations should be around 3-5 for the datasets used. After this number of recommendations, the recommendation performance starts to decline sharply. Thus, there should be a focus on quality rather than quantity.

6. Majority of personalization systems do not have a ranking component. Once, when *top* Υ recommendations are found out for a user, how these recommendations should be presented or in what order these recommendations should be given to the user, is very important to enhance a user's personnel experience. If highly preferred items are placed higher in the ranked list, user's personnel browsing experience is enhanced.

7. In cases when MDD techniques like tensors are used, our research discovered that in most of the cases choosing the best rank approximation for decomposition is a complex process, and often when higher ranks are used for decomposing the tensors, the performance of clustering starts to decline. Based on the many experiments conducted and evaluation of clustering on the result sets (using popular tensor methods and several rank decompositions - refer to Chapter 4), this thesis came up with an equation to determine the best decomposition rank for a tensor model. For a tensor defined as $\mathcal{T} \in \mathbb{R}^{M_1 \times M_2 \times .. \times M_n}$, let n be the total number of dimensions, then choosing the maximum upper limit of best rank decomposition(R_b) of $\mathcal{T}$, that will ensure the optimum clustering results based on various best fit adopted by each rank of the tensor decomposition can be defined as shown in Equation (58).

$$R_b = \frac{n+1}{2}. \tag{58}$$

8.3 Possible Future Work

With the increase in bandwidth and adoption of broadband plans by many countries, fast Internet speeds have becomes a reality. Due to this, the search for and exchange of images, videos, and other media files is becoming common. As an example, a user may watch videos of certain artists repeatedly or may watch certain categories of images. Identifying similar images, videos based on tags, content, ratings and a user's search behaviour history and then recommending him similar objects could be the future direction of research for Web personalization. The other direction of research is integrating the different profiles of a user (like a user may be a buyer as well as a seller). Identifying different profiles of a user within the same website is vital for adapting the recommender system to make appropriate recommendations to a relevant category of the user's profile. Relating searches in different domains; identifying knowledge from diverse sources; constructing methods of knowledge extraction from diverse data sources, and building complex user profiles and recommender systems, are some of the topics for future research into building multi-agent-based personalized systems that can cater to individual needs.

Another issue worth considering is that since a user's interest level changes with time, and each interest may range across a wide variety of domains, it is almost impossible to consider the breadth and depth of each interest in all domains. Therefore, this research has focussed solely on the automobile industry as its domain. All necessary guidance has been provided by the partner website, which has been taken as a prototype for conducting this research. The method of user profiling was restricted to the automobile sector, but similar models and methods could be employed in different domains based on their individual requirements.

Bibliography

[1] Acar, E. and Yener, B., "Unsupervised multiway data analysis: A literature survey," Technical report, Computer Science Department, Rensselaer Polytechnic Institute, Troy, NY. , 2007.

[2] Acar E., S. A. C. a., Mukkai S. Krishnamoorthy, and B¨ulent Yener, " Modeling and Multiway Analysis of Chatroom Tensors," in *Intelligence and Security Informatics*. vol. 3495/2005: Springer Berlin / Heidelberg, 2005, pp. 256-268.

[3] Adar, E., Weld, D. S., Bershad, B. N., and Gribble, S. S., "Why we search: visualizing and predicting user behavior," in *16th international conference on World Wide Web* Banff, Alberta, Canada: ACM Press New York, NY, USA, 2007, pp. 161-170.

[4] Adomavicius, G. and Tuzhilin, A., "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,* pp. 734-749, 2005.

[5] Aggarwal, C. C., Wolf, J. L., and Yu, P. S., "A new method for similarity indexing of market basket data," *ACM SIGMOD Record,* vol. 28, pp. 407-418, 1999.

[6] Agichtein, E., Brill, E., and Dumais, S., "Improving web search ranking by incorporating user behavior information," in *29th annual international ACM SIGIR conference on Research and development in information retrieval* Seattle, Washington, USA 2006, p. 26.

[7] Agrawal, R., and Srikant, R., "Fast algorithms for mining association rules," in *Proceedings of the 1994 International Conference on Very Large Databases*: 487-499, Santiago, Chile, 1994.

[8] Agrawal, S., Chaudhuri, S., Das, G., and Gionis, A., "Automated ranking of database query results," in *Proceedings of CIDR* 2003, pp. 171-183.

[9] Ahmad, A. and Dey, L., "A method to compute distance between two categorical values of same attribute in unsupervised learning for categorical data set," *Pattern Recognition Letters,* vol. 28, pp. 110-118, 2007.

[10] Albadvi, A. and Shahbazi, M., "A hybrid recommendation technique based on product category attributes," *Expert Systems With Applications,* vol. 36,Issue 9, pp. 11480-11488 2009.

[11] Amer-Yahia, S., Roy, S. B., Chawlat, A., Das, G., and Yu, C., "Group recommendation: Semantics and efficiency," *Proceedings of the VLDB Endowment,* vol. 2, pp. 754-765, 2009.

[12] Amigó, E., Gonzalo, J., Artiles, J., and Verdejo, F., "A comparison of extrinsic clustering evaluation metrics based on formal constraints," *Information Retrieval,* vol. 12, pp. 461-486, 2009.

[13] Andersen, C. M. and Bro, R., "Practical aspects of PARAFAC modeling of fluorescence excitation-emission data," *Journal of Chemometrics,* vol. 17, pp. 200-215, 2003.

[14] Anupriya Ankolekar, D. V., "Personalizing web surfing with semantically enriched personal profiles," in *Paper presented at International Workshop on Semantic Web Personalization* Budva Montenegro, 2006, pp. 1-73.

[15] Armstrong, R., Freitag, D., Joachims, T., and Mitchell, T., "Webwatcher: A learning apprentice for the world wide web," in *AAAI Spring Symposium on Information Gathering from Heterogeneous Distributed Environments*, 1995.

[16] Bader, B. W. and Kolda, T. G., "Efficient MATLAB computations with sparse and factored tensors," *SIAM Journal on Scientific Computing,* vol. 30, pp. 205-231, 2007.

[17] Balabanovi, M. and Shoham, Y., "Fab: content-based, collaborative recommendation," *Communications of the ACM,* vol. 40(3), pp. 66-72, 1997.

[18] Barbará, D., Li, Y., and Couto, J., "COOLCAT: an entropy-based algorithm for categorical clustering," in *11th International Conference on Information and knowledge Management* 2002, pp. 582-589.

[19] Barla, M., Tvarozek, M., and Bieliková, M., "Rule-based user characteristics acquisition from logs with semantics for personalized web-based systems," *Computing and Informatics,* vol. 28, pp. 399–427, 2009.

[20] Belkin, N. J. and Croft, W. B., "Information filtering and information retrieval: two sides of the same coin?," *Communications of the ACM,* vol. 35, pp. 29-38, 1992.

[21] Bell, R. M. and Koren, Y., "Scalable collaborative filtering with jointly derived neighborhood interpolation weights," in *Proc. of the 2007 Seventh IEEE Int. Conf. on Data Mining*, Washington, DC, USA, 2007, pp. 43-52.

[22] Billsus, D. and Pazzani, M. J., "A hybrid user model for news story classification," in *Proceedings of international conference on user modeling* Banff, Canada., 1999, pp. 99-108.

[23] Boriah, S., Chandola, V., and Kumar, V., "Similarity measures for categorical data: A comparative evaluation." vol. 30: Citeseer, 2007, p. 3.

[24] Breese, J. S., Heckerman, D., and Kadie, C., "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, Madison, WI, 1998, pp. 43–52.

[25] Bro, R., Harshman, R. A., Sidiropoulos, N. D., and Lundy, M. E., "Modeling multi way data with linearly dependent loadings," *Journal of Chemometrics,* vol. 23, pp. 324-340, 2009.

[26] Bro, R. and Kiers, H. A. L., "A new efficient method for determining the number of components in PARAFAC models," *Journal of Chemometrics,* vol. 17, pp. 274-286, 2003.

[27] Bueno, D. and David, A., "METIORE: A personalized information retrieval system," *User Modeling 2001,* pp. 168-177.

[28] Burke, R., "Knowledge-based recommender systems," *Encyclopedia of Library and Information Systems,* vol. 69, pp. 175-186, 2000.

[29] Burke, R., "Hybrid recommender systems: Survey and experiments," *User Modeling and User-Adapted Interaction,* vol. 12, pp. 331-370, 2002.

[30] Burke, R. D., Hammond, K. J., and Young, B. C., "The FindMe approach to assisted browsing," *IEEE Expert,* vol. 12, pp. 32-40, 1997.

[31] C. Stanfill., D. W., "Toward memory-based reasoning," *Communications of the ACM,* vol. 29, pp. 1213-1228, 1986.

[32] Carmel, D., Zwerdling, N., Guy, I., Ofek-Koifman, S., Har'el, N., Ronen, I., Uziel, E., Yogev, S., and Chernov, S., "Personalized social search based on the user's social network," in *CIKM*, Hong Kong, 2009, pp. 1227-1236.

[33] Carroll, J. D. and Chang, J. J., "Analysis of individual differences in multidimensional scaling via an N-way generalization of “Eckart-Young” decomposition," *Psychometrika,* vol. 35, pp. 283-319, 1970.

[34] Cattell, R. B., "“Parallel proportional profiles” and other principles for determining the choice of factors by rotation," *Psychometrika,* vol. 9, pp. 267-283, 1944.

[35] Chandra, P. a. W., Eric W., "Fibonacci Number," in *MathWorld--A Wolfram Web Resource.http://mathworld.wolfram.com/FibonacciNumber.html*

[36] Chang, C. C., Chen, P. L., Chiu, F. R., and Chen, Y. K., "Application of neural networks and Kano's method to content recommendation in web personalization," *Expert Systems with Applications,* vol. 36, pp. 5310-5316, 2009.

[37] Chaudhuri, S., Das, G., Hristidis, V., and Weikum, G., "Probabilistic ranking of database query results," in *Proceedings of the Thirtieth international conference on Very large data bases*, Toronto, Canada 2004, pp. 888 - 899

[38] Chaudhuri, S. and Gravano, L., "Evaluating top-k selection queries," in *Proceedings of the 25th VLDB Conference*, Edinburgh, Scotland, 1999, pp. 399-410.

[39] Chen, K. and Liu, L., "The" Best K" for entropy-based categorical data clustering," in *Proc of Scientific and Statistical Database Managemen t(SSDBM05)*, Santa Barbara, CA, June, 2005, pp. 253-262.

[40] Cho, Y. H. and Kim, J. K., "Application of Web usage mining and product taxonomy to collaborative recommendations in e-commerce," *Expert Systems with Applications,* vol. 26, pp. 233-246, 2004.

[41] Chou, P. H., Li, P. H., Chen, K. K., and Wu, M. J., "Integrating web mining and neural network for personalized e-commerce automatic service," *Expert Systems with Applications,* vol. 37, pp. 2898-2910, 2010.

[42] Chu, W. and Park, S. T., "Personalized recommendation on dynamic content using predictive bilinear models," in *International World Wide Web Conference*, Madrid, Spain, 2009, pp. 691-700.

[43] Daoud, M., Lechani, L. T., and Boughanem, M., "Towards a graph-based user profile modeling for a session-based personalized search," *Knowledge and Information Systems,* pp. 1-34, May 2009.

[44] Daoud, M., Tamine-Lechani, L., Boughanem, M., and Chebaro, B., "A session based personalized search using an ontological user profile," 2009, pp. 1732-1736.

[45] Das, G., Mannila, H., and Ronkainen, P., "Similarity of attributes by external probes," in *Fourth International Conference on Knowledge Discovery and Data Mining (KDD–98)*, 1998, pp. 23–29.

[46] De Almeida, A., Favier, G., and ao Mota, J., "The constrained block-PARAFAC decomposition," *TRICAP2006, Chania, Greece. Available online at http://www. telecom. tuc. gr/~ nikos/TRICAP2006main/TRICAP2006_Almeida. pdf,* 2006.

[47] De Lathauwer, L., De Moor, B., and Vandewalle, J., "A Multilinear Singular Value Decomposition," *SIAM JOURNAL ON MATRIX ANALYSIS AND APPLICATIONS,* vol. 21, pp. 1253-1278, 2000.

[48] Deshpande, M. and Karypis, G., "Item-based top-n recommendation algorithms," *ACM Transactions on Information Systems (TOIS),* vol. 22, pp. 143-177, 2004.

[49] Desrosiers, C. and Karypis, G., "A comprehensive survey of neighborhood-based recommendation methods," *Handbook on Recommender Systems, Springer,* 2009.

[50] Dou, Z., Song, R., and Wen, J. R., "A large-scale evaluation and analysis of personalized search strategies," in *16th international conference on World Wide Web* Banff, Alberta, Canada ACM Press New York, NY, USA, 2007, pp. 581-590.

[51] Douglas Carroll, J., Pruzansky, S., and Kruskal, J. B., "CANDELINC: A general approach to multidimensional analysis of many-way arrays with linear constraints on parameters," *Psychometrika,* vol. 45, pp. 3-24, 1980.

[52] Dunlavy, D. M., Kolda, T. G., and Kegelmeyer, W. P., "Multilinear algebra for analyzing data with multiple linkages," Technical Report SAND2006-2079, Sandia National Laboratories, Livermore, CA. http://csmr. ca. sandia. gov/~ tgkolda/pubs/SAND2006-2079. pdf 2006.

[53] Eirinaki, M., Mavroeidis, D., Tsatsaronis, G., and Vazirgiannis, M., "Introducing semantics in web personalization: The role of ontologies," *Semantics, Web and Mining,LNCS,* pp. 147-162, 2006.

[54] Escobar-Jeria, V., Martín-Bautista, M., Sánchez, D., and Vila, M. A., "Web Usage Mining Via Fuzzy Logic Techniques," *Foundations of Fuzzy Logic and Soft Computing,* pp. 243-252, 2007.

[55] Faloutsos, C., Kolda, T. G., and Sun, J., "Mining large graphs and streams using matrix and tensor tools," in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, Beijing , China, 2007, pp. 1174-1174.

[56] Frank, A. A., A., "UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]," Irvine, CA: University of California, School of Information and Computer Science, 2010.

[57] Fredman, M. L. and Tarjan, R. E., "Fibonacci heaps and their uses in improved network optimization algorithms." vol. 34: ACM, 1987, pp. 596-615.

[58] Ganti, V., Gehrke, J., and Ramakrishnan, R., "CACTUS—clustering categorical data using summaries," in *fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, San Diego, California, United States 1999, pp. 73-83.

[59] Gauch, S., Speretta, M., Chandramouli, A., and Micarelli, A., "User profiles for personalized information access," in *The Adaptive Web*: Springer, 2007, pp. 54-89.

[60] Germanakos, P., Tsianos, N., Lekkas, Z., Mourlas, C., Belk, M., and Samaras, G., "Embracing Cognitive Aspects in Web Personalization Environments--The AdaptiveWeb Architecture," 2007, pp. 430-431.

[61] Germanakos, P., Tsianos, N., Lekkas, Z., Mourlas, C., and Samaras, G., "Realizing Comprehensive User Profile as the Core Element of Adaptive and Personalized Communication Environments and Systems," *The Computer Journal,* 2008.

[62] Gibson, D., Kleinberg, J., and Raghavan, P., "Clustering categorical data: An approach based on dynamical systems," *The VLDB Journal,* vol. 8(3), pp. 222-236, 2000 2000.

[63] Godoy, D. and Amandi, A., "User profiling for web page filtering," *IEEE Internet Computing,* pp. 56-64, 2005.

[64] Godoy, D., Schiaffino, S., and Amandi, A., "Interface agents personalizing Web-based tasks," in *Cognitive Systems Research* vol. 5: Elsevier, 2004, pp. 207-222.

[65] Goel, M. and Sarkar, S., "Web site personalization using user profile information." vol. 2347/2006: LNCS,Springer, 2006, pp. 510-513.

[66] Goldberg, D., Nichols, D., Oki, B. M., and Terry, D., "Using collaborative filtering to weave an information tapestry," *Communications of the ACM,* vol. 35, pp. 61-70, 1992.

[67] Guha, S., Rastogi, R., and Shim, K., "Rock: A robust clustering algorithm for categorical attributes* 1," *Information Systems,* vol. 25, pp. 345-366, 2000.

[68] Harshman, R. A., "Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multi-modal factor analysis," *UCLA working papers in phonetics,* vol. 16, p. 84, 1970.

[69] Harshman, R. A., "PARAFAC2: Mathematical and technical notes," *UCLA working papers in phonetics,* vol. 22, pp. 30-47, 1972.

[70] Harshman, R. A. and Lundy, M. E., "Three-way DEDICOM: Analyzing multiple matrices of asymmetric relationships," in *Paper presented at the Annual Meeting of the North American Psychometric Society*, 1992.

[71] Harshman, R. A. and Lundy, M. E., "Uniqueness proof for a family of models sharing features of Tucker's three-mode factor analysis and PARAFAC/CANDECOMP," *Psychometrika,* vol. 61, pp. 133-154, 1996.

[72] Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T., "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems (TOIS),* vol. 22, pp. 5-53, 2004.

[73] Hill, W., Stead, L., Rosenstein, M., and Furnas, G., "Recommending and evaluating choices in a virtual community of use," in *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems,*, 1995, pp. 194-201.

[74] Holland, S., Ester, M., and Kießling, W., "Preference mining: A novel approach on mining user preferences for personalized applications," *Knowledge Discovery in Databases: PKDD 2003,* pp. 204-216, 2003.

[75] http://royal.pingdom.com/2010/01/22/internet-2009-in-numbers/.

[76] http://www.choicestream.com/surveyresults/.

[77] Huang, Z., "Extensions to the k-means algorithm for clustering large data sets with categorical values," *Data Mining and Knowledge Discovery,* vol. 2, pp. 283-304, 1998.

[78] Ian H. Witten, E. F., *Data Mining: Practical machine learning tools and techniques* vol. 2nd Edition: San Francisco: Morgan Kaufmann,, 2005.

[79] Ian H. Witten, E. F., *Data Mining: Practical machine learning tools and techniques, 2nd Edition* San Francisco: Morgan Kaufmann, 2005.

[80] Ichino, M., Yaguchi, H, "Generalized Minkoeski metrics for mixed feature-type data analysis," in *IEEE Transaction on Systems,Man and Cybernitics 24*, 1994, pp. 694–708.

[81] Ienco, D., Pensa, R., and Meo, R., "Context-Based Distance Learning for Categorical Data Clustering," Springer-Verlag Berlin Heidelberg, 2009, pp. 83-94.

[82] Iváncsy, R. and Vajk, I., "Frequent pattern mining in web log data," *Acta Polytechnica Hungarica,* vol. 3, pp. 77-90, 2006.

[83] Jansen, B. J., Booth, D. L., and Spink, A., "Patterns of query reformulation during Web searching," *Journal of the American Society for Information Science and Technology,* vol. 60, pp. 1358-1371, 2009.

[84] Jiang, X. and Tan, A. H., "Learning and inferencing in user ontology for personalized Semantic Web search," *Information Sciences,* vol. 179, pp. 2794-2808, 2009.

[85] Jin, X., Zhou, Y., and Mobasher, B., "A Unified Approach to Personalization Based on Probabilistic Latent Semantic Models of Web Usage and Content," in *AAAI 2004 Workshop on Semantic Web* San Jose, California,U.S.A., 2004.

[86] Joachims, T., Freitag, D., and Mitchell, T., "Webwatcher: A tour guide for the world wide web," in *Internaltional Joint Conference on Artificial Intelligence*, 1997, pp. 770-777.

[87] Jun, S. H., "Web usage mining using support vector machine," *Computational Intelligence and Bioinspired Systems,* pp. 349-356, 2005.

[88] Kaufman, L., Rousseeuw, P.J. , *Finding Groups in Data—An Introduction to Cluster Analysis*: Wiley, 1990.

[89] Kazienko, P., "Mining indirect association rules for web recommendation," *International Journal of Applied Mathematics and Computer Science,* vol. 19, pp. 165-186, 2009.

[90] Kazunari, S., Kenji, H., and Masatoshi, Y., "Adaptive web search based on user profile constructed without any effort from users," in *Proceedings of the 13th international conference on World Wide Web* New York, NY, USA: ACM Press, 2004.

[91] Kermarrec, A. M., "Challenges in personalizing and decentralizing the Web: an overview of GOSSPLE," *Stabilization, Safety, and Security of Distributed Systems,* pp. 1-16, 2009.

[92] Kießling, W. and Köstler, G., "Preference SQL: Design, implementation, experiences," in *Proceedings of the Very Large Database (VLDB) Conference*, Hong Kong, China, 2002, pp. 990-1001.

[93] Kim, H. N., Ji, A. T., Ha, I., and Jo, G. S., "Collaborative filtering based on collaborative tagging for enhancing the quality of recommendation," *Electronic Commerce Research and Applications,* vol. 9, pp. 73-83, 2010.

[94] Kodakateri Pudhiyaveetil, A., Gauch, S., Luong, H., and Eno, J., "Conceptual recommender system for CiteSeerX," in *RecSys*, New York, USA, 2009, pp. 241-244.

[95] Kolda, T. G., "Orthogonal Tensor Decompositions," *SIAM JOURNAL ON MATRIX ANALYSIS AND APPLICATIONS,* vol. 23, pp. 243-255, 2001.

[96] Kolda, T. G. and Bader, B. W., "Tensor decompositions and applications," Technical Report SAND2007-6702, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, , November 2007.

[97] Kolda, T. G., Bader, B. W., Kenny, J. P., Livermore, C. A., and Albuquerque, N. M., "Higher-Order Web Link Analysis Using Multilinear Algebra," in *Fifth IEEE International Conference on Data Mining (ICDM'05)* 2005, pp. 242-249.

[98] Kolda, T. G. and Sun, J., "Scalable tensor decompositions for multi-aspect data mining," in *ICDM*, Pisa 2008, pp. 363-372.

[99] Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., and Riedl, J., "GroupLens: applying collaborative filtering to Usenet news," *Communications of the ACM,* vol. 40, p. 87, 1997.

[100] Koren, Y., "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceeding of the 14th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, New York, USA, 2008, pp. 426-434.

[101] Kriegel., T. S. a. H.-P., "Optimal multi-step k-nearest neighbor search.," in *Proceedings of the 1998 ACM International Conference on Management of Data (SIGMOD'98)*, June 1998.

[102] Kroonenberg, P. M., *Three-mode principal component analysis: Theory and applications*: DSWO press, Leiden, The Netherlands, 1983.

[103] Kroonenberg, P. M. and De Leeuw, J., "Principal component analysis of three-mode data by means of alternating least squares algorithms," *Psychometrika,* vol. 45, pp. 69-97, 1980.

[104] Krulwich, B. and Burkey, C., "Learning user information interests through extraction of semantically significant phrases," in *Proceedings of the AAAI spring symposium on machine learning in information access*, Stanford, CA, 1996, pp. 100–112.

[105] Lacueva-Pérez, F. J., "Supervised Classification Fuzzy Growing Hierarchical SOM," in *Hybrid Artificial Intelligence Systems: Third International Workshop, HAIS*, 2008, p. 220.

[106] Lang, K., "Newsweeder: Learning to filter netnews," in *Proceedings of the twelfth international conference on machine learning*, Lake Tahoe, CA, 1995, pp. 331-339.

[107] Lazanas, A. and Karacapilidis, N., "Enhancing Recommendations through a Data Mining Algorithm," pp. 525-532.

[108] Lee, C. H., Kim, Y. H., and Rhee, P. K., "Web personalization expert with combining collaborative filtering and association rule mining technique," in *Expert Systems with Applications*. vol. 21: Elsevier, 2001, pp. 131-137.

[109] Leung, K. and Lee, D., "Dynamic Agglomerative-Divisive Clustering of Clickthrough Data for Collaborative Web Search," *LNCS,* pp. 635-642, 2010.

[110] Leung, K. W. T., Lee, D. L., and Lee, W. C., "Personalized Web search with location preferences," in *ICDE*, 2010, pp. 701-712.

[111] Li, G., Zhou, X., Feng, J., and Wang, J., "Progressive keyword search in relational databases," in *IEEE International Conference on Data Engineering*, 2009, pp. 1183-1186.

[112] Li, X., Zhang, J., and Li, L., "Providing Relevant Answers for Queries over E-Commerce Web Databases," in *Active Media Technology*: Springer Berlin / Heidelberg, 2009, pp. 467-477.

[113] Liao, H. and Ng, M. K., "Categorical data clustering with automatic selection of cluster number," *Fuzzy Information and Engineering,* vol. 1, pp. 5-25, 2009.

[114] Lieberman, H., "Letizia: An agent that assists web browsing," in *International Joint Conference on Artificial Intelligence* Montreal, 1995, pp. 924-929.

[115] Lieberman, H., Van Dyke, N., and Vivacqua, A., "Let's browse: a collaborative browsing agent," *Knowledge-Based Systems,* vol. 12, pp. 427-431, 1999.

[116] Linda Woolery, in *School of Nursing, University of Missouri* Columbia, MO 65211: Donor:Jerzy W. Grzymala-Busse (jerzy@cs.ukans.edu).

[117] Liu, A., Zhang, Y., and Li, J., "Personalized movie recommendation," in *ACM International Conference on Multi Media*, Beijing, China, 2009, pp. 845-848.

[118] Liu, D. R., Lai, C. H., and Lee, W. J., "A hybrid of sequential rules and collaborative filtering for product recommendation," *Information Sciences,* vol. 179, pp. 3505-3519, 2009.

[119] Liu, J. G., Zhou, T., Wang, B. H., Zhang, Y. C., and Guo, Q., "Effects of User's Tastes on Personalized Recommendation," *International Journal of Modern Physics C,* vol. 20, pp. 1925-1932, 2009.

[120] MacQueen, J. B., "Some methods for classification and analysis of multivariate observations.," in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, USA, 1967, pp. pp. 281–297.

[121] Maltz, D. and Ehrlich, K., "Pointing the way: Active collaborative filtering," in *Proceedings of the SIGCHI conference*, 1995, pp. 202-209.

[122] Martin-Bautista, M. J., Kraft, D. H., Vila, M. A., Chen, J., and Cruz, J., "User profiles and fuzzy logic for web retrieval issues," *Soft Computing-A Fusion of Foundations, Methodologies and Applications,* vol. 6, pp. 365-372, 2002.

[123] Matteo baldoni, C. b. a. N. H., ISI -Semantic web group University of hannover Germany, "Personalization for the Semantic Web," in *Reasoning Web*. vol. Volume 3564/2005: Springer Berlin / Heidelberg, 2005, pp. 173-212.

[124] Meng, X., Ma, Z. M., and Yan, L., "Answering approximate queries over autonomous web databases," in *Proceedings of the 18th international conference on World wide web*, Madrid, Spain 2009, pp. 1021-1030.

[125] Microsoft, C., "IIS Log File Format (IIS 6.0)," pp. http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/ Library/IIS/be22e074-72f8-46da-bb7e-e27877c85bca.mspx?mfr=true>.

[126] Min, S. H. and Han, I., "Detection of the customer time-variant pattern for improving recommender systems," *Expert Systems with Applications,* vol. 28, pp. 189-199, 2005.

[127] Mobasher, B., "Web Usage Mining and Personalization," in *Practical Handbook of Internet Computing, 2005*, M. P. Singh, Ed.: CRC Press LLC., 2005.

[128] Mobasher, B., "Data Mining for Web Personalization," in *The Adaptive Web*. vol. 4321, A. K. P. Brusilovsky, and W. Nejdl (Eds.), Ed., 2007, pp. 90-135.

[129] Mobasher, B., Cooley, R., and Srivastava, J., "Automatic personalization based on Web usage mining," *Communications of the ACM,* vol. 43, pp. 142-151, 2000.

[130] Mobasher, B., Dai, H., Luo, T., and Nakagawa, M., "Discovery of aggregate usage profiles for web personalization," *Data Mining and Knowledge Discovery,* vol. 6, pp. 61-82, 2002.

[131] Mushtaq, N., Werner, P., Tolle, K., and Zicari, R., "Building and evaluating non-obvious user profiles for visitors of Web sites," in *Proceedings on e-Commerce Technology, CEC*, 2004.

[132] Nasraoui, O. and Rojas, C., "From static to dynamic web usage mining: Towards scalable profiling and personalization with evolutionary computation," *Information Science,* vol. 178, pp. 3333-3346, 2003.

[133] Nasraoui, O., Soliman, M., Saka, E., Badia, A., and Germain, R., "A web usage mining framework for mining evolving user profiles in dynamic web sites," *IEEE Transactions on Knowledge and Data Engineering,* pp. 202-215, 2007.

[134] Pabarskaite, Z. and Raudys, A., "A process of knowledge discovery from web log data: Systematization and critical review." vol. 28: Springer, 2007, pp. 79-104.

[135] Park, Y. J. and Chang, K. N., "Individual and group behavior-based customer profile model for personalized product recommendation," *Expert Systems with Applications,* vol. 36, pp. 1932-1939, 2009.

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

[137] Petrovic, D., "Analysis of consumer Behaviour online," in *http://analogik.com/article_analysis_of_consumer_behaviour_online.asp*, 2007-08.

[138] Preda, M., Mirea, A. M., Teodorescu-Mihai, C., and Preda, D. L., "Adaptive Web Recommendation Systems," *Annals of the University of Craiova-Mathematics and Computer Science Series,* vol. 36, p. 25, 2009.

[139] Pretschner, A. and Gauch, S., "Ontology based personalized search," in *11th IEEE International Conference on Tools with Artificial Intelligence*, Chicago, 1999, pp. 391-398.

[140] Qian, G., Sural, S., Gu, Y., and Pramanik, S., "Similarity between euclidean and cosine angle distance for nearest neighbor queries," 2004, p. 1237.

[141] Rasmussen, M. and Karypis, G., "gcluto: An interactive clustering, visualization, and analysis system." vol. 21: Citeseer, 2008.

[142] Rawat, R., Nayak, R., and Li, Y., "Effective Hybrid Recommendation Combining Users-Searches Correlations Using Tensors," in *13th Asia-Pacific Web Conference* Beijing, China: Springer, 2011.

[143] Rawat, R., Nayak, R., and Li, Y., "Improving Web Database Search Incorporating Users Query Information," in *International Conference on Web Intelligence, Mining and Semantics-WIMS* Sogndal, Norway: ACM, 2011.

[144] Rawat, R., Nayak, R., and Li, Y., "Individual User Behaviour Modelling for Effective Web Recommendation," in *2nd International Conference on e-Education, e-Business, e-Management and e-Learing*, Mumbai, India, 2011, p. 5.

[145] Rawat, R., Nayak, R., and Li, Y., "Clustering of Web Users Using the Tensor Decomposed Models," in *Adjunct Proceedings of the 18th International Conference on User Modeling, Adaptation, and Personalization*, Big Island of Hawaii, HI, USA, June 20-24, 2010, p. 37.

[146] Rawat, R., Nayak, R., and Li, Y., "Clustering Web Users Using Tensor Space Modeling," in *WIMS-2011* Sogndal,Norway: ACM, May 2011.

[147] Rawat, R., Nayak, R., Li, Y., and Alsaleh, S., "Aggregate Distance Based Clustering Using Fibonacci Series-FIBCLUS," in *13th Asia-Pacific Web Conference*, Beijing ,China, 2011, p. 29—40.

[148] Rendle, S. and Marinho, B., "Learning optimal ranking with tensor factorization for tag recommendation," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, Paris, France 2009, pp. 727-736.

[149] Rendón, E. and Sánchez, J., "Clustering based on compressed data for categorical and mixed attributes." vol. 4109: Springer Berlin / Heidelberg, 2006, pp. 817-825.

[150] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J., "GroupLens: an open architecture for collaborative filtering of netnews," in *Proc. of the 1994 ACM Conference on Computer Supported Cooperative Work*, New York, 1994, pp. 175-186.

[151] Romero, C., Ventura, S., Zafra, A., and Bra, P., "Applying Web usage mining for personalizing hyperlinks in Web-based adaptive educational systems," *Computers & Education,* vol. 53, pp. 828-840, 2009.

[152] Rucker, J. and Polanco, M. J., "Personalized navigation for the web," *Communications of the ACM,* vol. 40, pp. 73-75, 1997.

[153] Ruxanda, P. S. M. and Manolopoulos, A. N. Y., "Ternary Semantic Analysis of Social Tags for Personalized Mucis Recommendation," in *Proceedings of the 9th International Conference on Music Information Retrieval*, 2008, p. 219.

[154] S.Q.Le. and T. B. Ho., "An association-based dissimilarity measure for categorical data," *Pattern Recognition Letters,* vol. 26, pp. 2549-2557, 2005.

[155] Salton, G. and McGill, M. J., *Introduction to modern information retrieval*: McGraw-Hill, 1983.

[156] San, O. M., Huynh, V. N., and Nakamori, Y., "An alternative extension of the k-means algorithm for clustering categorical data," in *Internation Journal of Applied Mathematics and Computer Science*. vol. 14: University of Zielona Gora Press, 2004, pp. 241-248.

[157] Sarwar, B., Karypis, G., Konstan, J., and Reidl, J., "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International WWW Conference*, Hong Kong, 2001, pp. 285-295.

[158] Sarwar, B., Karypis, G., Konstan, J., Riedl, J., and Minnesota Univ Minneapolis Dept Of Computer, S., *Application of dimensionality reduction in recommender system-a case study*: Citeseer, 2000.

[159] Savas, B. and Eldén, L., "Handwritten digit classification using higher order singular value decomposition," *Pattern Recognition,* vol. 40, pp. 993-1003, 2007.

[160] Schafer, J. B., Konstan, J., and Riedi, J., "Recommender systems in e-commerce," in *Proceedings of the First ACM Conference on Electronic Commerce*, Denver,USA, 1999, pp. 158-166.

[161] Shardanand, U. and Maes, P., "Social information filtering: algorithms for automating “word of mouth”," in *Proc. of the SIGCHI Conf. on Human factors in Computing Systems*, New York, 1995, pp. 210-217.

[162] Sheehan, B. N. and Saad, Y., "Higher order orthogonal iteration of tensors (hooi) and its relation to pca and glram," SIAM, 2007, pp. 355–366.

[163] Silverstein, C., Marais, H., Henzinger, M., and Moricz, M., "Analysis of a very large web search engine query log." vol. 33: ACM New York, NY, USA, 1999, pp. 6-12.

[164] Sinha, R. and Swearingen, K., "Comparing recommendations made by online systems and friends," in *Proceedings of Delos-NSF Workshop on Personalisation and Recommender Systems in Digital Libraries*, 2001.

[165] Skillicorn, D., *Understanding complex datasets: data mining with matrix decompositions*: Chapman & Hall/CRC, 2007.

[166] Soliman, M. A. and Ilyas, I. F., "Ranking with uncertain scores," in *IEEE International Conference on Data Engineering*, 2009, pp. 317–328.

[167] Srivastava, J., Cooley, R., Deshpande, M., and Tan, P. N., "Web usage mining: discovery and applications of usage patterns from Web data." vol. 1: ACM Press New York, NY, USA, 2000, pp. 12-23.

[168] Stermsek, G., Strembeck, M., and Neumann, G., "A user profile derivation approach based on log-file analysis," in *Proceedings of the International Conference on Information and Knowledge Engineering (IKE)*, Las Vegas, NV, USA, June,2007.

[169] Stewart, A., Niederée, C., and Mehta, B., "State of the Art in User Modelling for Personalization in Content, Service and Interaction," *NSF/DELOS Report on Personalization,* 2004.
[170] Summers, S., "School of Nursing, University of Kansas Medical Center," Kansas City, KS 66160: Donor:Jerzy W. Grzymala-Busse (jerzy@cs.ukans.edu)
[171] Sun, J., Tao, D., and Faloutsos, C., "Beyond streams and graphs: dynamic tensor analysis," in *12th ACM SIGKDD international conference on Knowledge discovery and data mining*, Philadelphia, PA, USA, 2006, pp. 374-383.
[172] Sun, J. T., Zeng, H. J., Liu, H., Lu, Y., and Chen, Z., "CubeSVD: a novel approach to personalized Web search," in *International World Wide Web Conference Committee (IW3C2)*, Chiba, Japan., May 10-14,2005, pp. 382-390.
[173] Symeonidis, P., Nanopoulos, A., and Manolopoulos, Y., "Tag recommendations based on tensor dimensionality reduction," in *RecSys '08'*, Lausanne, Switzerland, 2008, pp. 43-50.
[174] Takács, G., Pilászy, I., Németh, B., and Tikk, D., "Major components of the gravity recommendation system," *ACM SIGKDD Explorations Newsletter,* vol. 9, pp. 80-83, 2007.
[175] Tao, D., Song, M., Li, X., Shen, J., Sun, J., Wu, X., Faloutsos, C., and Maybank, S. J., "Bayesian tensor approach for 3-D face modeling," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 18, pp. 1397-1410, 2008.
[176] Terveen, L., Hill, W., Amento, B., McDonald, D., and Creter, J., "PHOAKS: A system for sharing recommendations," *Communications of the ACM,* vol. 40, pp. 59-62, 1997.
[177] Tucker, L. R., "Some mathematical notes on three-mode factor analysis," *Psychometrika,* vol. 31, pp. 279-311, 1966.
[178] Ucar, D., Altiparmak, F., Ferhatosmanoglu, H., and Parthasarathy, S., "Investigating the use of extrinsic similarity measures for microarray analysis," in *Proceedings of the BIOKDD workshop at the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.
[179] Ungar, L. H. and Foster, D. P., "Clustering methods for collaborative filtering," in *Proceedings of the AAAI98 Workshop on Recommendation Systems*, Madison Wisconsin, 1998, pp. 112–125.
[180] US eCommerce Forecast: 2008 To 2012, F. R., January 18, 2008.
[181] Wang, X., Abraham, A., and Smith, K. A., "Intelligent web traffic mining and analysis," *Journal of Network and Computer Applications,* vol. 28, pp. 147-165, 2005.
[182] Wei, C., Sen, W., Yuan, Z., and Lian-Chang, C., "Algorithm of mining sequential patterns for web personalization services," *ACM SIGMIS Database,* vol. 40, pp. 57-66, 2009.
[183] Weng, S. S. and Liu, M. J., "Feature-based recommendations for one-to-one marketing," *Expert Systems with Applications,* vol. 26, pp. 493-508, 2004.
[184] Wittenburg, K., Das, D., Hill, W., and Stead, L., "Group asynchronous browsing on the World Wide Web," in *Proceedings of the Fourth International World Wide Web Conference*, Boston, MA, 1995, pp. 51-62.
[185] Woolery, L., Grzymala-Busse, J., Summers, S., and Budihardjo, A., "The use of machine learning program LERS-LB 2.5 in knowledge acquisition for expert system development in nursing." vol. 9, 1991, p. 227.
[186] www.internetworldstats.com/stats.htm.
[187] Xiubo, G., Tie-Yan, L., Tao, Q., Andrew, A., Hang, L., and Heung-Yeung, S., "Query dependent ranking using K-nearest neighbor," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval* Singapore, Singapore: ACM, 2008.
[188] Xu, Y., Ishikawa, Y., and Guan, J., "Effective top-k keyword search in relational databases considering query semantics," in *Advances in Web and Network Technologies, and Information Management*. vol. 5731/2010: Springer Berlin / Heidelberg, 2010, pp. 172-184.
[189] Yu, Y., Lin, H., Yu, Y., and Chen, C., "Personalized Web Recommendation Based on Path Clustering," *Flexible Query Answering Systems,* pp. 368-377, 2006.
[190] Zhang, T., Ramakrishnan, R., and Livny, M., "BIRCH: an efficient data clustering method for very large databases," in *ACM SIGMOD, International Conference on Management of Data*, 1996, pp. 103-114.
[191] Zhang, Z. K. and Zhou, T., "Effective Personalized Recommendation in Collaborative Tagging Systems," *Arxiv preprint arXiv:0907.3315,* 2009.
[192] Zhou, B., Hui, S. C., and Fong, A. C. M., "Web Usage Mining for Semantic Web Personalization," in *PerSWeb'05 workshop on Personalization on the Semantic* Edinburgh, UK, 2005.

[193] Zhou, B., Hui, S. C., and Fong, A. C. M., "An Effective Approach for Periodic Web Personalization," in *2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)*, Hong Kong, 2006, pp. 284-292.

[194] Zhu, L., Shen-Da Ji, W., and Liu, C. N, "Keyword Search Based on Knowledge Base in Relational Databases," in *8th Intl. Conf. on Machine Learning and Cybernetics*, 2009.

[195] Zhuhadar, L., Nasraoui, O., and Wyatt, R., "Dual representation of the semantic user profile for personalized web search in an evolving domain," in *AProceedings of the AAAI 2009 Spring Symposium on Social Semantic Web, Where Web 2.0 meets Web 3.0*, 2009, pp. 84–89.