A HYBRID FILTERING APPROACH FOR RECOMMENDER SYSTEMS USING CLUSTERING FOR CHAINS

A PROJECT REPORT

Submitted by

PRIYESH.V (21908104067)
SIVARANJANI.S (21908104089)
VIMAL RAJ.S (21909104111)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

SRI VENKATESWARA COLLEGE OF ENGINEERING
PENNALUR, SRIPERUMBUDUR 602 105

ANNA UNIVERSITY: CHENNAI 600 025

MAY, 2013

ANNA UNIVERSITY CHENNAI: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"A HYBRID FILTERING APPROACH FOR RECOMMENDER SYSTEMS USING CLUSTERING FOR CHAINS"** is the bonafide work of **"PRIYESH.V, SIVARANJANI.S, VIMAL RAJ.S"** who carried out the project work under my supervision.

SIGNATURE

Dr. T. K. THIVAKARAN
HEAD OF THE DEPARTMENT

Professor
Department of Computer
Science and Engineering,
Sri Venkateswara College of
Engineering,
Sriperumbudur,
Chennai – 602 105

SIGNATURE

Ms. B. SATHIYA
SUPERVISOR

Assistant Professor
Department of Computer
Science and Engineering,
Sri Venkateswara College of
Engineering,
Sriperumbudur,
Chennai – 602 105

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

Recommender systems (RS) are widely used tools and techniques for personalization in information retrieval systems. The task of RS is to generate recommendation lists for users over unrated items. Traditional recommender systems perform the task of predicting preferences over unrated items based on rating scores rather than rankings. Learning to rank (LOR) is an established means of predicting rankings for recommender systems. The problem with LOR technique is the high complexity of the rank aggregation problem. In this project we propose a Hybrid Social Recommender system which is a content boosted collaborative filtering technique using Clustering for Chains. The adaptation of Clustering chains for recommender systems improves the computational complexity of LOR problem. We address the problems of cold start and data sparstiy with our novel trust based social content boosting technique. The result of the proposed work is 28% better than traditional collaborative filtering methods.

ACKNOWLEDGEMENT

We thank our principle **Dr. M. Sivanandham, Ph.D.,** Sri Venkateswara College of Engineering for being the source of inspiration throughout our study in this college.

We express our sincere thanks to **Dr. T. K. Thivakaran, Ph.D.,** Head of the Department, **Computer Science and Engineering** for his permission and encouragement accorded to carry this project.

With profound respect, we express our deep sense of gratitude and sincere thanks to our guide, **Ms. B. Sathiya, M. Tech.,** Assistant Professor, for her valuable guidance and suggestions throughout this project.

We express our sincere thanks to **Mr. S. Shivashankar**, **Project Head, Ericson R & D**, Chennai for mentoring us throughout this project.

We are also thankful to **Mrs. S. Kalavathi, M.E.,** Assistant Professor and Project coordinator, for her continual support and assistance.

We thank our family and friends for all their support and encouragement throughout the course of our graduate studies.

PRIYESH.V

SIVARANJANI.S

VIMAL RAJ.S

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

IR	Information retrieval
IN	Invited and not attending
NIN	Not invited , not attending
INR	Invited, not responded
NINR	Not invited, not responded
IY	Invited, attending
NIY	Not invited, attending
UEMAT	User Event Matrix
UCMAT	User Cluster Matrix
CBF	Content Based Filtering
RS	Recommender Systems

CHAPTER 1

INTRODUCTION

1.1 INFORMATION RETRIEVAL

The concept of "information retrieval" is broad and often employed in an imprecise meaning. It is used to denote systems designed to provide users or a group of users with information. Salton/McGill (1983:x1) give this definition: "An information retrieval system is an information system, that is, a system used to store items of information that need to be processed, searched, retrieved and disseminated to various user populations."

1.2 SKETCH OF AN INFORMATION RETRIEVAL SYSTEM

IR is divided into two parts namely updating and retrieval. The updating consists of the subsystem preparing data for retrieval. The objective is to have the data represented in the system in such a way that it may efficiently be exploited in the retrieval process. The updating consists of two sub processes:

- Preparing data (text) for retrieval (adding value).

- Storing of data.

The first process will be to transform information on the object to be retrieved (and the object may be a book or a text) to data. This will generally imply an analysis of the information and representation of this as physical symbols with a structure corresponding to the one given for the retrieval system. In a document retrieval system one will frequently, from practical or economic reasons, select a representation of the source different from the authentic text - for instance an abstract or a set of indexing terms. The document design, for instance the intellectual indexing, is such a transformation of source to document.

Preparing data for retrieval often takes place manually, but in some systems there are automatic routines for some of this activity. Such a computerization implies that it is possible to specify exactly how the data is to be processed, and this may be quite difficult. Much research has been aimed at developing efficient and advanced methods for automatic indexing, but few of these methods have been implemented (cfr the work of Salton). During the last years, the work within the area of artificial intelligence has opened new possibilities for such work.

The storage includes a systematization of data in such a way that it may be retrieved fast and efficiently. In a manual information retrieval system one would, for instance, sort indexing terms alphabetically, while a

computerized system has a number of more sophisticated possibilities. In a document retrieval system, one will generally store each word in an inverted file, and associate each word with references to the documents in which the word occurs. The retrieval process can be made more efficient by creating an index to the inverted file. The retrieval process is the part of the system most characteristic of an information retrieval system. Its objective is to retrieve the documents required by the user - and these only. This subsystem consists of three major processes.

The first step in a retrieval process is the construction of search requests. The user transforms his problem into a form (the request) which is compatible to the data base design and the requirements of the retrieval system. It is important for the retrieval performance that the request corresponds to the way in which the information is represented in the retrieval system. If there is a lack of correspondence, it will be difficult - or impossible - to achieve a satisfactory result. The retrieval itself consists of matching the search request against the data base, i.e retrieving a certain telephone number from the directory, or all documents containing a certain phrase. The way in which the result is represented to the user will primarily depend on the type of information retrieval system employed. In a manual system, the result is in a fixed format, while a computerized system may give a choice of different formats and responses. Several systems also have the possibility to rank documents according to criteria for possible relevance.

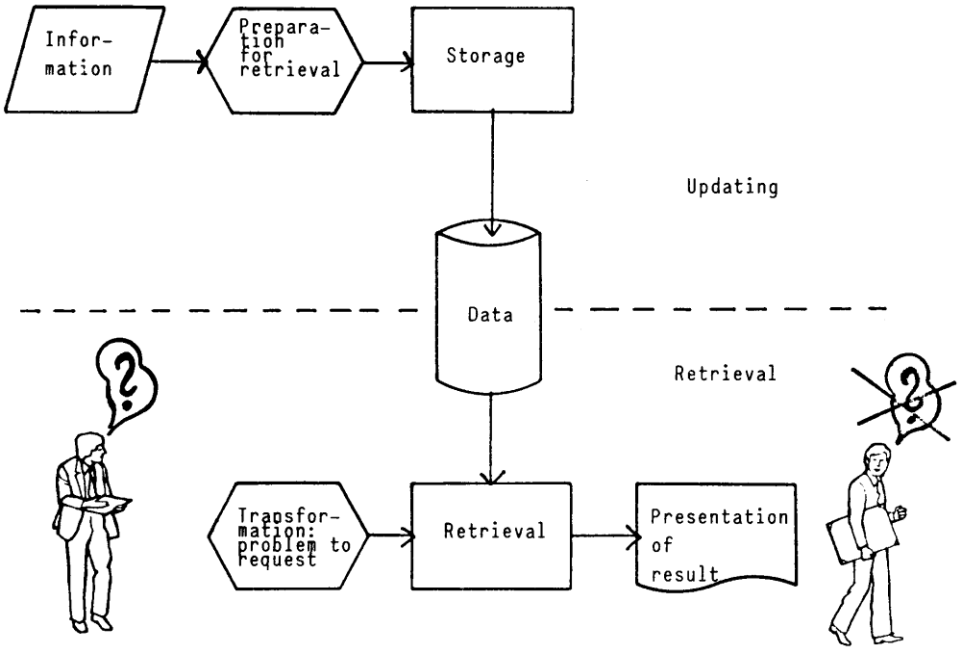

Figure 1.1 Sketch of an IR System

1.3 LEARNING TO RANK FOR IR

With huge amounts of information available to us at hand, all of these information become baseless without proper retrieval. Ranking now comes into the picture. Ranking is used in a variety of applications like Document Retrieval, Collaborative Filtering, Key Term extraction, important email routing, sentimental analysis, product rating, anti-web spam etc. Rank the documents purely according to their relevance with regards to the query.For example, Consider the relationships of similarity, website structure, and diversity between documents in the ranking process (relational ranking).Aggregate several candidate ranked lists to get a better ranked list (meta search) etc.

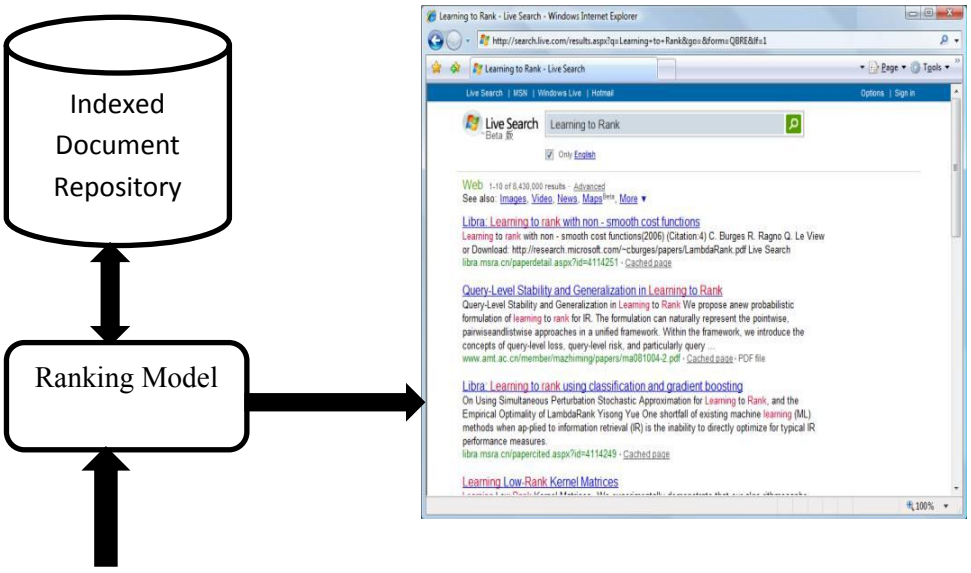

Figure 1.2 Importance of Ranking

In general, those methods that use machine learning technologies to solve the problem of ranking can be named as "learning to rank" methods. Eg: Personalized Query search engine of eBay.

1.4 MACHINE LEARNING

Machine Learning is programming computers to optimize a performance criterion using example data or past experience. We have a model defined up to some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data or past experience. The model can be predictive to make predictions in

the future, or descriptive to gain knowledge from data, or both. As per Tom Mitchell (Mitchell (1997)), "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks T, as measured by P, improves with experience E".Machine learning algorithms can be organized into a taxonomy based on the desired outcome of the algorithm or the type of input available during training the machine.

- Supervised learning generates a function that maps inputs to desired outputs (also called labels, because they are often provided by human experts labeling the training examples). For example, in a classification problem, the learner approximates a function mapping a vector into classes by looking at input-output examples of the function.

- Unsupervised learning models a set of inputs, like clustering. See also data mining and knowledge discovery. Here, labels are not known during training.

- Semi-supervised learning combines both labeled and unlabeled examples to generate an appropriate function or classifier. Transduction, or transductive inference, tries to predict new outputs on specific and fixed (test) cases from observed, specific (training) cases.

- Reinforcement learning learns how to act given an observation of the world. Every action has some impact in the environment, and the environment provides feedback in the form of rewards that guides the learning algorithm. Learning to learn learns its own inductive bias based on previous experience.

1.5 CLUSTERING

Clustering can be considered the most important unsupervised learning problem; so as every other problem of this kind, it deals with finding a structure in a collection of unlabeled data. A cluster is therefore a collection of objects which are “similar” between them and are “dissimilar” to the objects belonging to other clusters.

For example,

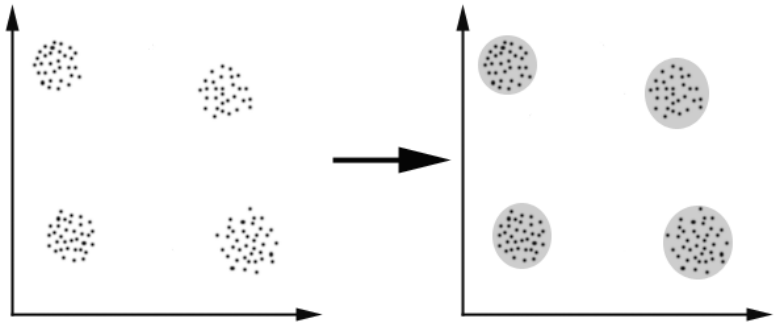

Figure 1.3 Clustering

In this case we easily identify the 4 clusters into which the data can be divided; the similarity criterion is distance: two or more objects belong to

the same cluster if they are “close” according to a given distance (in this case geometrical distance).

1.5.1 K Means Clustering

K Means is one of the simplest unsupervised algorithms that solve the well-known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed apriori. The main idea is to define k centers, one for each cluster. These centers should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest center. When no point is pending, the first step is completed and an early group age is done. At this point we need to recalculate k new centroids as barycenter of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest newcenter. A loop has been generated. As a result of this loop we may notice that the k centers change their location step by step until no more changes are done or in other words centers do not move any more. Finally, this algorithm aims at minimizing an objective function know as squared error function given by Equation 1.1.

$$J(V)= \sum_{i=1}^{c_i} \sum_{j=1}^{c_j} (||x_i - v_j||)^2 \qquad \text{Equation (1.1)}$$

Where, '$||x_i - v_j||$' is the Euclidean distance between x_i and v_j.

'c_i' is the number of data points in i^{th} cluster.

'c' is the number of cluster centers.

Let $X = \{x_1,x_2,x_3,\ldots\ldots,x_n\}$ be the set of data points and $V = \{v_1,v_2,\ldots\ldots,v_c\}$ be the set of centers.

1. Randomly select 'c' cluster centers.
2. Calculate the distance between each data point and cluster centers.
3. Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers.
4. Recalculate the new cluster center .
5. Recalculate the distance between each data point and new obtained cluster centers.
6. If no data point was reassigned then stop, otherwise repeat from step 3.

The following shows how the K means work for five items and two clusters.

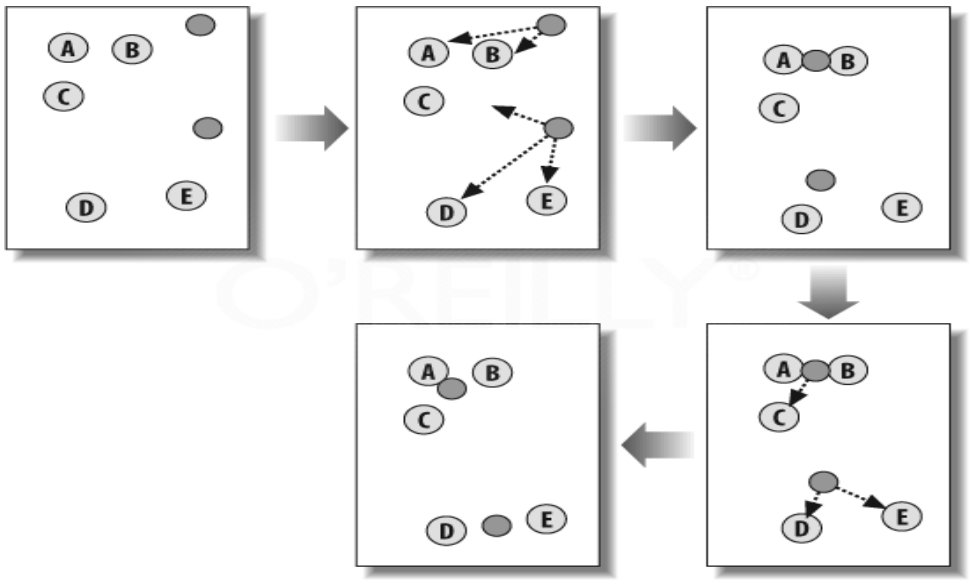

Figure 1.4 K Means Example

Even though K Means is Fast, robust and easier to understand and relatively efficient: O(tknd), where n is no. of objects, k is no. of clusters, d is no. of dimension of each object, and t is no. of iterations, it has its own share of disadvantages. For these reasons, we adapt the K Means algorithm in order to perform clustering of chains and use the same in recommender systems.

1.6 RECOMMENDER SYSTEMS

Recommender systems or recommendation systems (sometimes replacing "system" with a synonym such as platform or engine) are a subclass

of information filtering system that seek to predict the 'rating' or 'preference' that user would give to an item (such as music, books, or movies) or social element (e.g. people or groups) they had not yet considered, using a model built from the characteristics of an item (content-based approaches) or the user's social environment (collaborative filtering approaches)**.**Recommender systems have become extremely common in recent years.

1.6.1 Examples of Recommender Systems

When viewing a product on Amazon.com, the store will recommend additional items based on a matrix of what other shoppers bought along with the currently selected item. Like many E-commerce sites, Amazon.com™ (www.amazon.com) is structured with an information page for each book, giving details of the text and purchase information. The Customers who Bought feature is found on the information page for each book in their catalog. It is in fact two separate recommendation lists. The first recommends books frequently purchased by customers who purchased the selected book. The second recommends authors whose books are frequently purchased by customers who purchased works by the author of the selected book.

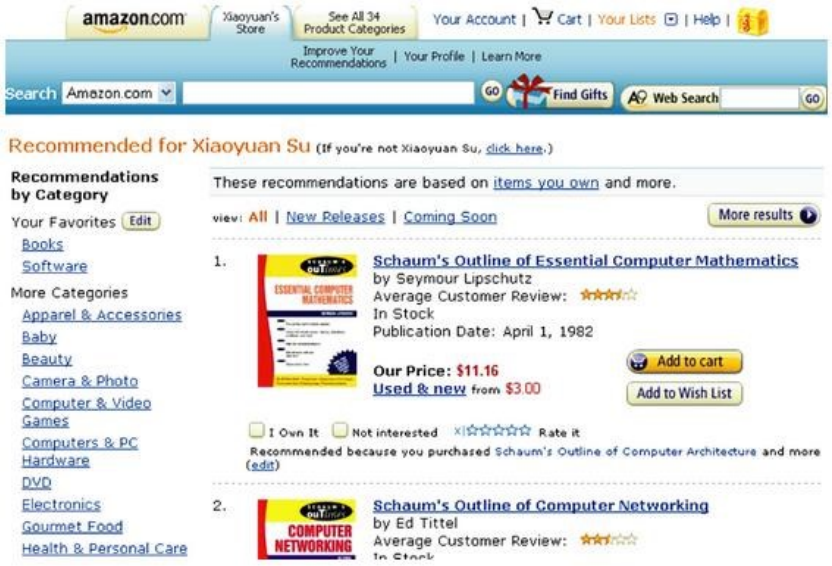

Figure 1.5 Amazon's Recommender System

1.6.1.1 eBay

Feedback Profile: The Feedback Profile feature at eBay.com™ (www.ebay.com) allows both buyers and sellers to contribute to feedback profiles of other customers with whom they have done business. The feedback consists of a satisfaction rating (satisfied/neutral/dissatisfied) as well as a specific comment about the other customer. Feedback is used to provide a recommender system for purchasers, who are able to view the profile of sellers. This profile consists of a table of the number of each rating in the past 7 days, past month, and past 6 months, as well as an overall summary (e.g., 867 positives from 776 unique customers). Upon further request, customers can browse the individual ratings and comments for sellers.

1.6.1.2 Netflix

Netflix offers predictions of movies that a user might like to watch based on the user's previous ratings and watching habits (as compared to the behavior of other users), also taking into account the characteristics (such as the genre) of the film.

1.6.1.3 Reel.com

Movie Matches: Similar to Amazon.com ,Reel.com's Movie Matches (www.reel.com) provides recommendations on the information page for each movie. These recommendations consist of "close matches" and/or "creative matches." Each set consists of up to a dozen hyperlinks to the information pages for each of these "matched" films. The hyperlinks are annotated with one sentence descriptions of how the new movie is similar to the original movie in question ("Darker thriller raises similarly disturbing questions…").

Movie Map: The Movie Map feature of Reel.com recommends movies to customers based on syntactic features. Customers enter queries based on Genre, movie types, viewing format and/or prices, and request results be constrained to "sleepers" or "best of this genre." The recommendations are editor's recommendations for movies that fit the specified criteria.

1.6.2 Collaborative Filtering

One approach to the design of recommender systems that has seen wide use is collaborative filtering. Collaborative filtering methods are based on collecting and analyzing a large amount of information on users' behaviors, activities or preferences and predicting what users will like based on their similarity to other users. A key advantage of the collaborative filtering approach is that it does not rely on machine analyzable content and therefore it is capable of accurately recommending complex items such as movies without requiring an "understanding" of the item itself. Many algorithms have been used in measuring user similarity or item similarity in recommender systems. For example, the k-nearest neighborhood (k-NN) approach and the Pearson Correlation. When building a model from a user's profile, a distinction is often made between explicit and implicit forms of data collection .Examples of explicit data collection include the following:

- Asking a user to rate an item on a sliding scale.
- Asking a user to rank a collection of items from favorite to least favorite.
- Presenting two items to a user and asking him/her to choose the better one of them.
- Asking a user to create a list of items that he/she likes.

Examples of implicit data collection include the following:

- Observing the items that a user views in an online store.
- Analyzing item/user viewing times
- Keeping a record of the items that a user purchases online.
- Obtaining a list of items that a user has listened to or watched on his/her computer.
- Analyzing the user's social network and discovering similar likes and dislikes.

The recommender system compares the collected data to similar and dissimilar data collected from others and calculates a list of recommended items for the user. Several commercial and non-commercial examples are listed in the article on collaborative filtering systems. One of the most famous examples of collaborative filtering is item-to-item collaborative filtering (people who buy x also buy y), an algorithm popularized by Amazon.com's recommender system. Other examples include the following,

- **Last.fm** recommends music based on a comparison of the listening habits of similar users.

- **Facebook, MySpace, LinkedIn**, and other social networks use collaborative filtering to recommend new friends, groups, and other social connections (by examining the network of connections between a user and their friends).

Collaborative filtering approaches often suffer from three problems: cold start, scalability, and sparsity.

• **Cold Start**: These systems often require a large amount of existing data on a user in order to make accurate recommendations.

• **Scalability**: In many of the environments that these systems make recommendations in, there are millions of users and products. Thus, a large amount of computation power is often necessary to calculate recommendations.

• **Sparsity**: The number of items sold on major e-commerce sites is extremely large. The most active users will only have rated a small subset of the overall database. Thus, even the most popular items have very few ratings. A particular type of collaborative filtering algorithm uses matrix factorization, a low-rank matrix approximation technique.

1.6.3 Content Based Filtering

Another common approach when designing recommender systems is content-based filtering. Content-based filtering methods are based on information about and characteristics of the items that are going to be recommended. In other words, these algorithms try to recommend items that are similar to those that a user liked in the past (or is examining in the

present). In particular, various candidate items are compared with items previously rated by the user and the best-matching items are recommended. This approach has its roots in information retrieval and information filtering research.

Basically, these methods use an item profile (i.e., a set of discrete attributes and features) characterizing the item within the system. The system creates a content-based profile of users based on a weighted vector of item features. The weights denote the importance of each feature to the user and can be computed from individually rated content vectors using a variety of techniques. Simple approaches use the average values of the rated item vector while other sophisticated methods use machine learning techniques such as Bayesian Classifiers, cluster analysis, decision trees, and artificial neural networks in order to estimate the probability that the user is going to like the item.

Direct feedback from a user, usually in the form of a like or dislike button, can be used to assign higher or lower weights on the importance of certain attributes (using Rocchio Classification or other similar techniques).

A key issue with content-based filtering is whether the system is able to learn user preferences from user's actions regarding one content source and use them across other content types. When the system is limited to

recommending content of the same type as the user is already using, the value from the recommendation system is significantly less than when other content types from other services can be recommended. For example, recommending news articles based on browsing of news is useful, but it's much more useful when music, videos, products, discussions etc. from different services can be recommended based on news browsing. As previously detailed, Pandora Radio is a popular example of a content-based recommender system that plays music with similar characteristics to that of a song provided by the user as an initial seed. There are also a large number of content-based recommender systems aimed at providing movie recommendations, a few such examples include Rotten Tomatoes, Internet Movie Database, Jinni, Rovi Corporation and See This Next.

CHAPTER 2

RECOMMENDER SYSTEMS

2.1 INTRODUCTION

In general, every recommendation system follows a specific process in order to produce product recommendations. The recommendation approaches can be classified based on the information sources they use. Three possible sources of information can be identified as input for the recommendation process. The available sources are the user data (demographics), the item data (keywords, genres) and the user-item ratings (obtained by transaction data, explicit ratings).

Basically, there is a set of users and a set of items. Each user *u* rates a set of items by some values. The task of a recommender system is to predict the rating of user *u* on an un-rated item *I* or recommend some items for user *u* based on the existing ratings. Techniques in RS can be divided into three categories: collaborative filtering, content-based recommendation and hybrid approaches. Collaborative filtering make recommendations based on the ratings of item *I* by the set of users whose rating profiles are most similar to

that of user *u*. Content-based methods use the features of items, e.g. movie's genres, directors, actors, etc., to generate recommendations. Hybrid approaches make recommendations by combining collaborative filtering and content-based recommendation.

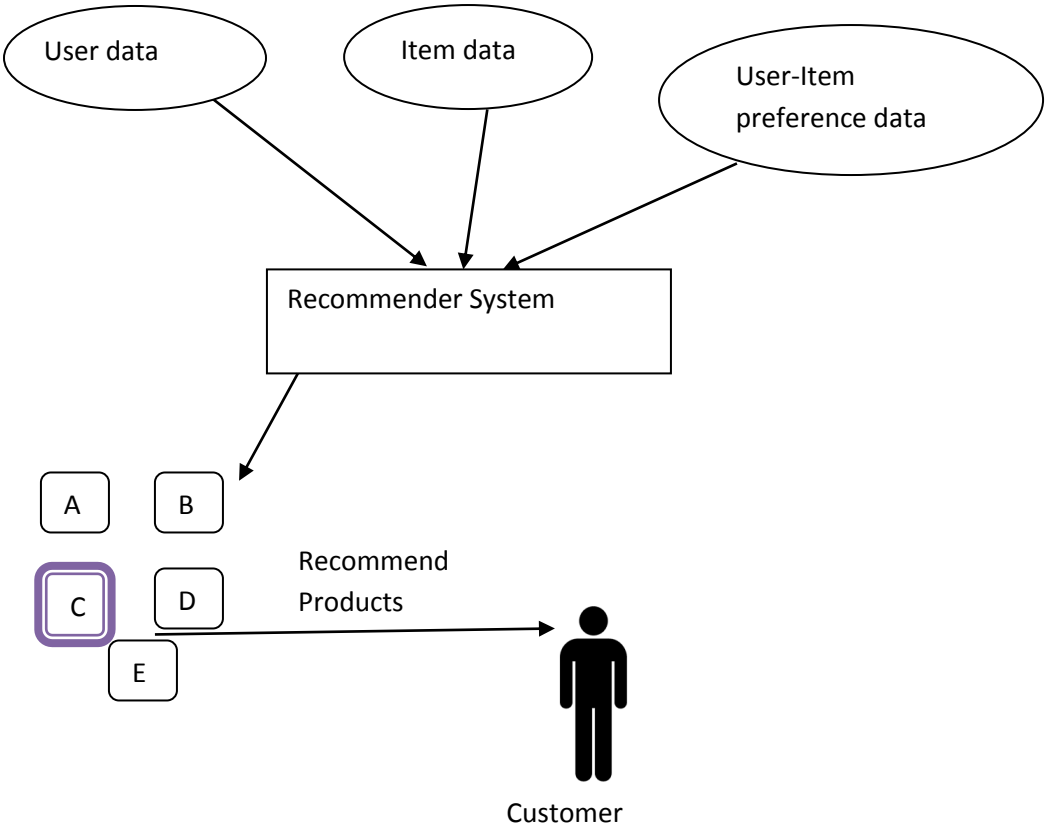

Figure 2.1 Recommender Process

2.2 DATA

The recommender systems would not be functional without data. Some recommender systems need data about the user, some need data about the products and some recommender systems need both. Data can be provided by the customer explicitly or implicitly. There are many ways to acquire data.

2.2.1 Explicit user data

Explicit data is given by a customer. A rating can be given on a particular scale, this is often a scale of five stars, where one star represents the lowest ranking and five stars the highest ranking. Scales of more than five stars are possible as well. Amazon.com uses the following meaning to the stars: 1 star: I hate it; 2 stars: I don't like it; 3 stars: It's OK; 4 stars: I like it; 5 stars: I love it.

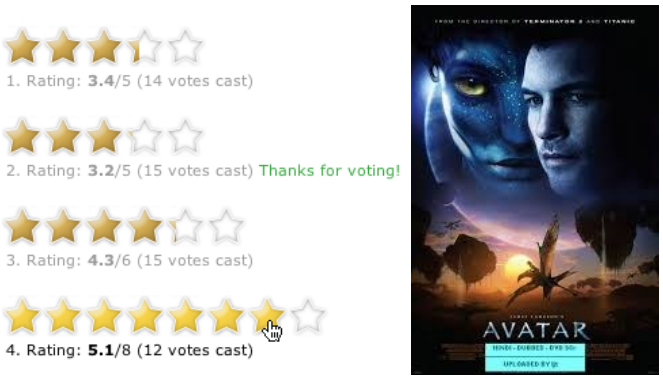

Figure 2.2 Rating of Movies

Then prepare a user-item rating matrix as in Figure 2.3

User\Rating	**I1**	**I2**	**I3**					**Im**
U1	R(1,1)	R(1,2)						
U2								
U3								R(3,m)

Figure 2.3 User-item Rating Matrix

Where U1, U2, .., Un are the n users that use the particular system. I1, I2,...Im are the m items that can be rated by the users. And R(k,j) represent the rating of user k for item j.

2.2.2 Implicit Data

Not all users rate all the items they have bought or viewed, because they just do not want to spend their time rating the items or do not see the point of doing so. And not all customers register to the shop and want to give all their personal information. For these users another information source is needed to overcome the lack of ratings. One approach to this problem is to use implicit ratings: watching the behavior of the user.

2.3 DIFFERENT APPROACHES TO RECOMMENDER SYSTEMS

The data captured is used by the recommender system to eventually provide the recommendations to the customer. Recommender systems can be present in all sorts of systems and situations, and thus can be implemented in many different ways.

1. Non-personalized
2. Demographic-based
3. Collaborative filtering
4. Content-based
5. Knowledge-based
6. Hybrid

We will focus on Collaborative, Content Based and Hybrid approaches as we use a combination of these for our proposed architecture.

2.3.1 Collaborative filtering

Collaborative filtering methods are based on collecting and analyzing a large amount of information on users' behaviors, activities or preferences and predicting what users will like based on their similarity to other users. A key advantage of the collaborative filtering approach is that it does not rely on machine analyzable content and therefore it is capable of accurately

recommending complex items such as movies without requiring an "understanding" of the item itself.

The collaborative filtering approach can be divided in two categories:

1. Memory-based
2. Model-based

2.3.1.1 Memory based

Memory-based Collaborative Filtering algorithms use the entire or a sample of the user-item rating matrix to generate a prediction. Every user is part of a group of people with similar interests. By identifying the so-called neighbors of an active user, a prediction of preferences on items for him or her can be produced. In the neighborhood-based algorithm (User-based Collaborative Filtering), a subset of users are chosen based on their similarity to the active user, and a weighted combination of their ratings is used to produce predictions for this user. Most of these approaches can be generalized by the algorithm summarized in the following steps:

1. Assign a weight to all users with respect to similarity with the active user.

2. Select k users that have the highest similarity with the active user (neighbors).

3. Compute a prediction from a weighted combination of the selected neighbors' ratings.

The similarity computation: Step 1 is a critical step in memory-based CF algorithms. The weight $w_{a,u}$ is a measure of similarity between the user u and the active user a. There are many different methods to compute similarity between users. The most commonly used measure of similarity is the Pearson correlation coefficient between the ratings of the two users:

$$w_{a,u} = \frac{\sum_{i \in I} (r_{a,i} - \bar{r}_a)(r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i \in I} (r_{a,i} - \bar{r}_a)^2 \sum_{i \in I} (r_{u,i} - \bar{r}_u)^2}} \qquad \text{Equation (2.1)}$$

Where I is the set of items rated by both users, $r_{a,i}$ is the rating given to item i by the active user a, $r_{u,i}$ is the rating given to item i by user u, and $\bar{r}_u$ is the mean rating given by user. In step 3 predictions are generally computed as the weighted average of deviations from the neighbor's mean, as in:

$$p_{a,i} = \bar{r}_a + \frac{\sum_{u \in K} (r_{u,i} - \bar{r}_u) \times w_{a,u}}{\sum_{u \in K} w_{a,u}} \qquad \text{Equation (2.2)}$$

In Equation 2.2, $p_{a,i}$ is the prediction for the active user a for item i, $w_{a,u}$ is the similarity between users a and u, and K is the set of most similar users.

The *vector-based similarity* is another way to compute similarity. It is originally used to calculate the similarity between two documents by treating each document as a vector of word frequencies and computing the cosine of the angle formed by the frequency vectors. This formalism can be adopted in collaborative filtering, which uses the ratings of two users as a vector in an m-dimensional space, and compute similarity based on the cosine of the angle between them, given by Equation 2.3.

$$w_{a,u} = \cos(\vec{r}_a, \vec{r}_u) = \frac{\vec{r}_a \cdot \vec{r}_u}{\|\vec{r}_a\|_2 \times \|\vec{r}_u\|_2} = \frac{\sum_{i=1}^{m} r_{a,i} r_{u,i}}{\sqrt{\sum_{i=1}^{m} r_{a,i}^2} \sqrt{\sum_{i=1}^{m} r_{u,i}^2}} \quad \text{Equation (2.3)}$$

There have been several other similarity measures used in the literature, including Spearman rank correlation, Kendall's τ correlation, mean squared differences, entropy, and adjusted cosine similarity.

The *User-based CF* approach does not scale well when it is applies to millions of users and items, because of the computational complexity of the search for similar users.

Item-based Collaborative Filtering where they match a users' rated items to similar items, rather than matching similar users. In this approach, similarities between pairs of items i and j are computed off-line using Pearson correlation is given by Equation 2.4.

$$w_{i,j} = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_j)^2}}, \qquad \text{Equation (2.4)}$$

where $r_{u,i}$ is the rating of user u on item i, U is the set of all users who rated both items i and j, $\bar{i}$ is the average rating of the ith item by those users. See Figure 2.4, where we can see that user 2, l and n co- rated the same items i and j .

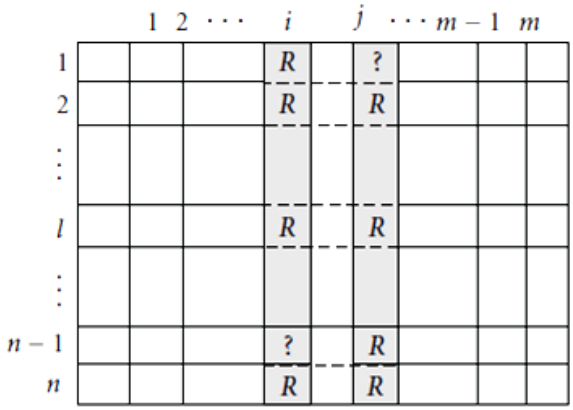

Figure 2.4 Item-based Similarity ($w_{i,j}$) Calculation

Now, the rating for item i for user a can be predicted using a simple weighted average, as in Equation 2.5.

$$p_{a,i} = \frac{\sum_{j \in K} r_{a,j} w_{i,j}}{\sum_{j \in K} |w_{i,j}|} \qquad \text{Equation (2.5)}$$

Where K is the neighbour-hood set of the k items rated by a that are most similar to i. For item-based collaborative filtering too, one may use alternative similarity metrics such as adjusted cosine similarity.

2.3.1.2 Model based

Model-based recommendation systems involve building a model based on the dataset of ratings. In other words, information has to be extracted from the dataset, and can be used a "model" to make recommendations without having to use the complete dataset every time. This approach potentially offers the benefits of both speed and scalability.

Model-based collaborative filtering algorithms include Bayesian models (probabilistic), clustering models and recently, a new method based on matrix factorization is the most promising approach now.

From a probabilistic perspective, the collaborative filtering can be viewed as calculating the expected value of a rating, given the profile of the

user or the previous ratings. Assume that the ratings are integers with a range for 0 to m, the probability that the active user will have a particular rating for item *j* given the previously observed ratings is then calculated.

2.3.1.3 Hybrid

Burke did a survey on several hybrid approaches. Hybrid recommender system is another category of recommender systems that tries to overcome the limitations of the other approaches. A Hybrid recommender system combines two or more recommendation techniques to gain better system optimization and fewer of the weaknesses of any individual ones. The most popular hybrid approaches are those of content-based and collaborative filtering**.** There are different strategies by which hybridization can be achieved and they are broadly classified into seven categories as shown in Table 2.1.

Table 2.1 Hybrid Recommendation

Hybridization	**Description**
Weighted	The ratings of several recommendation techniques are combined together to produce a single recommendation
Switching	The system switches between recommendation techniques depending on the current
Mixed	Recommendations from several different recommenders are presented at the same time
Feature combination	Features from different recommendation data sources are thrown together into a single recommendation algorithm
Cascade	One recommender refines the recommendations given by another
Feature augmentation	Output from one technique is used as an input feature to another
Meta-level	The model learned by one recommender is used as input to another

CHAPTER 3

LITERATURE SURVEY

3.1 CLUSTERING OF CHAINS

A chain is a totally ordered subset of a finite set of items. Chains are an intuitive way to express preferences over a set of alternatives, as well as a useful representation of ratings in situations where the item-specific scores are either difficult to obtain, too noisy due to measurement error, or simply not as relevant as the order that they induce over the items. Informally, chains are totally ordered subsets of a set of items, meaning that for all items that belong to a chain we know the order, and for items not belonging to the chain the order is unknown. For example, consider a preference survey about movies where the respondents are requested to rank movies they have seen from best to worst. In this scenario chains are a natural representation for the preference statements, as it is very unlikely that everyone would list the same movies. In a clustering of the responses people with similar preferences should be placed in the same cluster, while people who strongly disagree should be placed in different clusters.

Formally,

- Consider M to be a set of m items, $\pi \subseteq M$
- (u,v) belongs to π means that u precedes v according to π for items in $M \setminus \pi$, π does not specify the order. This is called Partial Order.
- If π is defined over entire set of M items it is called Total Order.

Linear Extension- In order theory, a branch of mathematics, a linear extension of a partial order is a linear order (or total order) that is compatible with the partial order. It may be viewed as an order-preserving bijection from a partially ordered set P to a chain C on the same ground set. Mathematically, Linear Extension of a partial order π is a total order tau so that $(u,v) \in \pi \rightarrow (u,v) \in \tau$.In Ukkonen's (2011) [3] paper, Lloyd's algorithm (Lloyd, 1982)is adapted for chains. The main problem is the lack of a good distance function for chains, as well as the computational complexity of rank aggregation. At the core of our approach is to consider the probabilities of pairs of items to precede one another in the cluster. He also presents two methods for mapping chains to high-dimensional vector spaces. The first method aims to preserve the distance between two chains that are assumed to originate from the same component in a simple generative model. The second method represents each chain as the mean of the set of linear extensions of the chain. He has also introduced a new theorem stating that this can be achieved with a very simple

mapping. In particular, it is not necessary to enumerate the set of linear extensions of a chain.

An algorithm is also devised for uniformly sampling sets of chains that share a number of characteristics with a given set of chains. The random sets of chains are used for significance testing. Finally they conduct a number of experiments to compare the proposed method with existing algorithms for clustering chains and it turns out that the algorithms are in some sense orthogonal. For smaller data sets the algorithms by Kamishima and Akaho (2009) [2] give in most cases a better result. However, as the input size increases, the method proposed in this paper outperforms other algorithms.

3.2 SOCIAL NETWORK AND RECOMMENDATIONS

3.2.1 Social Trust

In the real-world, when looking for a recommendation of a restaurant or movie, we often turn to our friends, on the basis that we have similar food or movie preferences. However, a particular friend may not be reliable when it comes to recommending a particular type of movie. For example, you may ask Alice to suggest a romantic-comedy, but you would not ask her to recommend an action-thriller movie. There is a measurement of trustworthiness which we have in our friends' recommendations. This suggests that similarity alone will not always provide the most reliable

recommendation partners. Recommendation partners should have similar tastes and preferences, but they should also be trust worthy. Social trust is very complex and depends on many factors, which makes is difficult to model in a computational system. Some factors which influence trust are: past experience with a person, relationship with the person, opinions of the actions a person has taken, psychological factors impacted by a lifetime of history and events, rumor, and influence by others' opinions .Much work has been done to formalize the concept of social trust into computing environments. Social influence plays an important role in product marketing. However, it has rarely been considered in traditional recommender systems.

J.He et al (2005) [7] have presented a new paradigm of recommender systems which can utilize information in social networks, including user preferences, item's general acceptance, and influence from social friends. A probabilistic model is developed to make personalized recommendations from such information. We extract data from a real online social network, and our analysis of this large dataset reveals that friends have a tendency to select the same items and give similar ratings. Experimental results on this dataset show that our proposed system not only improves the prediction accuracy of recommender systems but also remedies the data sparsity and cold-start issues inherent in collaborative filtering. Furthermore, they have proposed to improve the performance of our system by applying semantic filtering of social networks, and validate its improvement via a class project experiment. In this experiment we demonstrate how relevant friends

can be selected for inference based on the semantics of friend relationships and finer-grained user ratings.

3.2.2 Existing Work

There has been an increasing amount of research being performed in the area of trust based recommender systems; however, it is still in the early stages of development. Golbeck et al (2005) [6] applies the concept of trust to Web-based social networks. They describe how trust can be computed and how it can be used in applications. Specifically, they propose two algorithms for inferring trust relationships between individuals who are not directly connected to the network. The researchers apply their technique to the Trust Mail application, which is an email client that uses the trust algorithms to sort email messages in the user's inbox depending on the ratings in the trust network.

Pitsilis et al (2003) [8] explored trust in decentralized recommender systems. They developed a model which uses quantitative and qualitative parameters to build trust relationships between entities based on their common choices. Trust propagation was used to extend trust relationships beyond the direct neighbors. Their recommender system was tested on various lengths, or "hops," of trust propagation. The experimental results showed a significant decrease in data sparseness and prediction error.

Massa et al (2007) [4] introduced a trust-aware recommendation architecture which relies on users explicitly stating trust values for other users. They also use a trust propagation technique applied to the network of users, to estimate a trust weight that can be used in place of the similarity weight. An evaluation of their system on the Epinions.com dataset reveals that their system is successful in lowering the mean error on predictive accuracy for cold start users. The authors also define a trade-off situation between recommendation coverage and accuracy in the system.

O'Donovan et al (2005) [9] propose a system which is similar to this work. They present algorithms for calculating a profile-level trust and item-level trust. Their trust metrics compute the percentage of correct recommendations that the user has contributed. The item-level trust represents the percentage of times that a user correctly recommended a particular item. Their experimental results show a positive impact on the overall prediction error rates.

CHAPTER 4

PROPOSED ARCHITECTURE AND IMPLEMENTATION

4.1 PROPOSED ARCHITECTURE

The overall architecture of the proposed model is as follows:

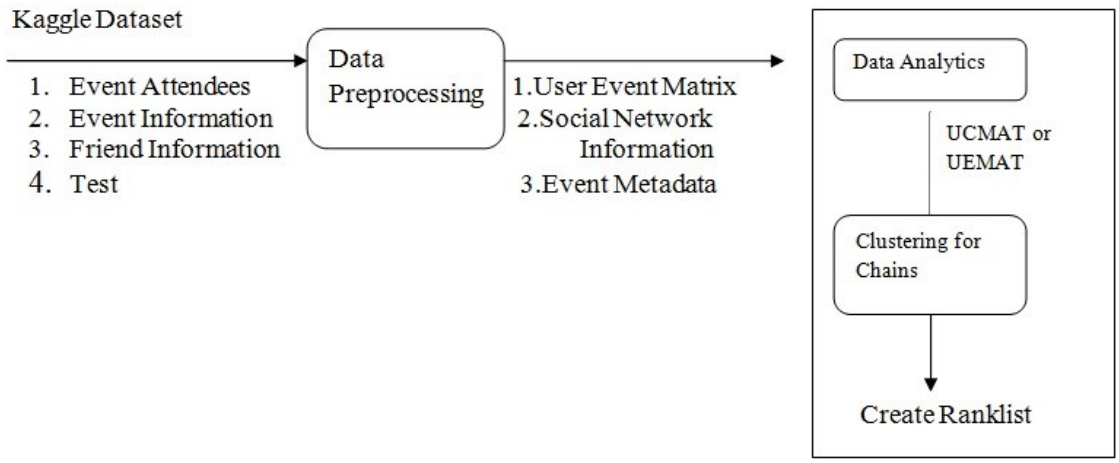

Figure 4.1 Proposed Architecture

The architecture is roughly divided into three stages namely Data Pre-processing, Data Analytics, and Clustering for Chains. The data sets

which are to be used for pre-processing are downloaded from a website called Kaggle which conducts an online event recommendation challenge. These data sets are then used by the data pre-processing module. The outcome of this, yields three important pieces of information namely user event matrix, social network information and event metadata. The next two modules fall under the process of hybrid filtering approach for recommendation which combines data analytics and clustering for chains. A rank list is then created. Each of the modules above are described in detail in the following sections.

4.2 THE KAGGLE CHALLENGE

Kaggle is a platform for predictive modelling and analytics competitions on which companies and researchers post their data and statisticians and data miners from all over the world compete to produce the best models. This crowd sourcing approach relies on the fact that there are countless strategies that can be applied to any predictive modelling task and it is impossible to know at the outset which technique or analyst will be most effective. Kaggle conducts an event recommendation challenge which asks to predict what events users will be interested in based on events they have responded to in the past, user demographic information, and what events they have seen and clicked on in Kaggle's app. The insights discovered from this data, and the algorithms the participants create, allows Kaggle to improve their event recommendation algorithm and enrich user experience. They also have two types of leader boards namely the Public Leader Board and Private Leader Board. The public leader board is used when the challenge is ongoing

and participants want to test their results whereas the private leader board is used when there is no on-going challenge, but simply for research and learning purposes. Their data set is described in detail in the following sections.

4.3 THE KAGGLE DATASET

The following datasets are used and they are in the form of CSV or Comma Separated Value files.

1. **Test.csv** contains the same columns as train.csv, except for interested and not_interested. Each row corresponds to an event that was shown to a user in our application. **event** is an id identifying an event in a our system. **user** is an id representing a user in our system. **invited** is a binary variable indicated whether the user has been invited to the event. **interested** is a binary variable indicating whether a user clicked on the "Interested" button for this event; it is 1 if the user clicked Interested and 0 if the user did not click the button. Similarly, **not_interested** is a binary variable indicating whether a user clicked on the "Not Interested" button for this event; it is 1 if the user clicked the button and 0 if not. It is possible that the user saw an event and clicked neither Interested nor Not Interested, and hence there are rows that contain 0,0 as values for **interested,not_interested**.

2. **user_friends.csv** contains social data about this user, and contains two columns: user and friends. **user** is the user's id in our system, and **friends** is a space-delimited list of the user's friends' ids.

3. **events.csv** contains data about events in our system, and has 110 columns. **event_id**, **user_id**, the columns**. event_id** is the id of the event, and**user_id** is the id of the user who created the event. The last 101 columns require a bit more explanation; first, we determined the 100 most common word stems (obtained via Porter Stemming) occuring in the name or description of a large random subset of our events. The last 100 columns are **count_1**, **count_2**, ..., **count_100**, where **count_N**is an integer representing the number of times the Nth most common word stem appears in the name or description of this event.

4. **event_attendees.csv** contains information about which users attended various events, and has the following columns such as: **event_id**, **yes**, **maybe**, **invited,** and **no. event_id i**dentifies the event. **yes**, **maybe**,**invited**, and **no** are space-delimited lists of user id's representing users who indicated that they were going, maybe going, invited to, or not going to the event.

In summary the data sets and their specifications are as follows :

Table 4.1 Data Sets

DATA SET	SIZE	SPECIFICATIONS
Event attendees	115 MB	24805 Event attendees
Events	1.09 GB	30 lakh events
User-Friends	311 MB	39752 users
Test	575 KB	1357 users

4.3.1 Evaluation Metrics

Suppose there are m missing outbound edges from a user in a social graph, and you can predict up to 10 other nodes that the user is likely to follow. Then, by adapting the definition of average precision in IR), the average precision at n for this user is:

$$ap@n=\sum_{k=1}^{n} P(k)/min(m,10) \qquad \text{Equation (4.1)}$$

where if the denominator is zero, the result is set zero; P(k) means the precision at cut-off k in the item list, i.e., the ratio of number of users followed up to the position k over the number k, and P(k) equals 0 when k -th item is not followed upon recommendation; n = 10

(1) If the user follows recommended nodes #1 and #3 along with another node that wasn't recommend, then ap@10 = (1/1 + 2/3)/3 ≈ 0.56

(2) If the user follows recommended nodes #1 and #2 along with another node that wasn't recommend, then ap@10 = (1/1 + 2/2)/3 ≈ 0.67

(3) If the user follows recommended nodes #1 and #3 and has no other missing nodes, then ap@10 = (1/1 + 2/3)/2 ≈ 0.83

The mean average precision for N users at position n is the average of the average precision of each user, i.e.,

$$MAP@n=\sum_{i=1}^{N} ap@n_i/N \qquad \text{Equation (4.2)}$$

Note this means that order matters: if recommend two nodes A & B in that order and a user follows node A and not node B, MAP@2 score will be higher (better) than if recommended for B and then A.

4.4 DATA PREPROCESSING

The data preprocessing phase consists of the following stages namely metadata preparation, test set preparation and preparing user event matrix.

4.4.1 Metadata Preparation

For metadata preparation we consider the Event file which consists of Event_id, Owner_id and 100 keywords metadata. We put these information into a MAT file namely Event.MAT which has over 30 lakhs users.

4.4.2 Test Set Preparation

Here we prepare a test set consisting of users and the events. There are 1357 users in consideration and 2328 events in total. Out of these 1357 users we consider only 590 users. Out of these 590 users 226 have some recorded event history meaning they have responded to some events. The rest 364 users have no event history at all but only just friends' information. We need to predict a ranking list for these users right from most interested to least interested for around five to six events.

4.4.3 Prepare User Event Matrix

The next step would be to prepare a user event matrix or UEMAT. There are 1357 users and these users have twelve lakh friends. The contents of the matrix have to be filled in the following way.

IN	NIN	INR	NINR	MAYBE	IY	NIY
-3	-2	-1	0	1	2	3

IN	=	Invited and not attending
NIN	=	Not invited and not attending
INR	=	Invited, not responded
NINR	=	Not invited and not responded
IY	=	Invited and attending
NIY	=	Not invited and attending

The next step would be to filter this matrix according to the number of users and the number of events to which they have responded. We then get a matrix the order of 350000 X 20000 meaning 3.5 lakh users and 20000 events. The final step would be to filter according to event metadata, so in the end the order of the matrix is 3,44,628 X 19196 meaning 3.44 lakh users and 19196 events.

4.4.4 Preparing Friends Information

In this section, a Friends Adjacency matrix is prepared and friendship and trust are calculated subsequently. These are roughly based on two categories namely Non trust based and trust based. Trust is defined as the level to which one user believes or is connected to another user. Trust is increased if a person have large number of mutual friends or has similar preferences or taste compared to another user.

4.4.4.1 Non Trust Based

If a user and another user are connected to each other but they don't have mutual friends nor similar tastes/preferences friends information is prepared by non-trust based method where an average is taken of all events attended by a particular user and his friends.

4.4.4.2 Trust Based

Two methods are proposed here. The first one is trust by mutual friends and the next one trust by similarity preferences.

4.4.4.2.1 Mutual Friends

This works on the concept that the more the mutual friends a user has with another one, the more the trust is bound to increase. For example if A has 15 mutual friends with B and 45 mutual friends with C, we can very well say that A trusts C more than B.

4.4.4.2.2 Preference Similarity

This works on the concept of having similar tastes/preferences between two users. If A and B have responded for the same two events in a similar manner their trust level is likely to be more than A and C who have responded for four common events, where C has responded in a way totally opposite to that of A. The pseudocode for the algorithm is as follows:

```
fori=1 to maximum user events
    get_user_eventlist();
    get_user_friendslist();
        for j=1 to maximum number of friends
        for k=1 to maximum user events
            check for the number of common events
                calculate dissimilarity=((count_commonevents)/
                (maximum_user_events)*6))
                Calculate friendship=1-(abs(dissimilarity)*100)
```

Algorithm 4.1: Calculating Trust by Preference Similarity

4.5 DATA ANALYTICS

We are preparing a setup which helps us choose the best boosting techniques and best boosting setup. The major part of this module involves content based filtering after which comes the boosting technique.

4.5.1 Content Based Filtering

As derived from UEMAT we have 19000 events in total. These events are then clustered using K Means algorithm to get the maximum possible clusters. Trying out the different possibilities, it is found that the maximum possible clusters got here is 200 beyond which empty clusters start occurring. The general content based approach proposed here is as follows,

Feature Normalization: featurenormalize(X) returns a normalized version of X where the mean value of each feature is 0 and the standard deviation is 1. This is often a good preprocessing step to do when working with learning algorithms.

K Means Clustering: Here K means clustering is done to cluster events according to categories. The parameters are X, K, uniform, Distance, Correlation, MaxIter, 1000. Here X is the data set, K the initial number of clusters, MaxIter indicates the maximum iterations to be run which is specified here to be 1000.

User Cluster Matrix (UCMAT): The user cluster matrix is created in this step. The next step to be found out is how many events occur for each category of clusters. Then those events are taken and probability is calculated for preferred response for these events. Both these probabilities are divided to get UCMAT.

The pseudocode is as follows:

```
f = find(UEMAT(i,:)); % retrieve User Event History
    for j = 1:size(f,2)
        calculate count of events responded in each category of clusters
    if UEMAT(i,f(1,j)) > 0
        calculate count of events which are preferred by users
    end
    end
    for k = 1:K
        divide the counts to obtain the probability
    End
```

Algorithm 4.2 Creating User Cluster Matrix

4.5.2 Boosting Techniques

As outlined previously we have three boosting techniques proposed which fall broadly into two categories namely Trusted and Non trusted techniques. By now we have two kinds of information which are user information and friends information. This is going to be the input for the boosting techniques. The general form is as follows:

User Event value= (C_1 *User Value) + (C_2* Friends Value).

where C_1 and C_2 are constants.

For example,

If we take C_1=1 and C_2=0 this is the case of partial boosting where we completely ignore friends information and consider only user information. If we take C_1=0 and C_2=1, this is called Non equal weightage boosting where the user information is completely ignored and friends information is only considered. The last case is where both C_1=0.5 and C_2=0.5 and it is referred to as equal weightage boosting.

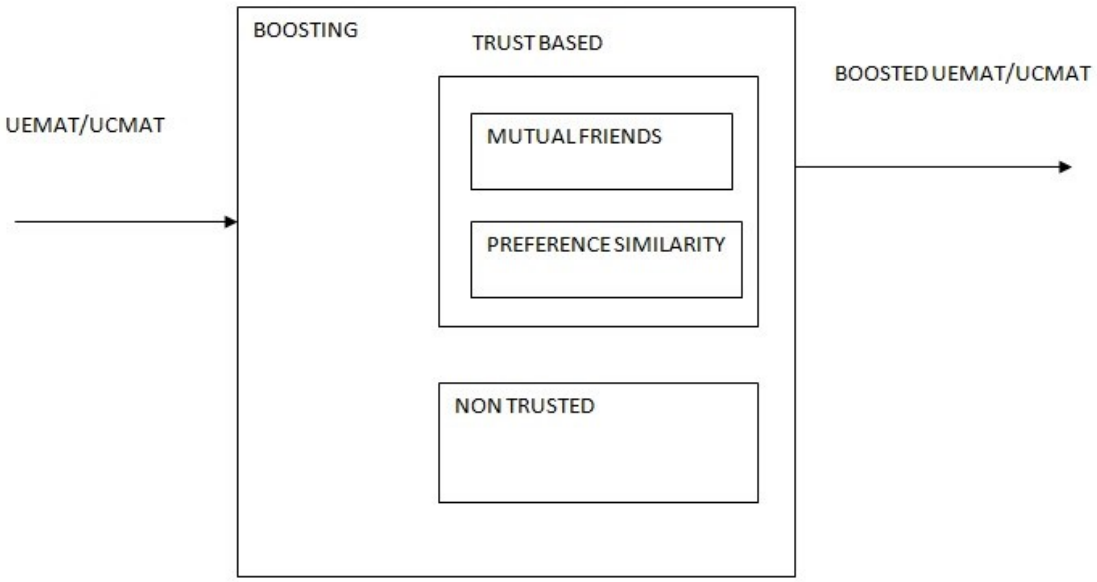

Figure 4.2 Boosting

4.5.2.1 Boosting By Mutual Friends

This takes into consideration that more the number of mutual friends greater the friendship and thereby trust. The following is the algorithm for this technique. First both event list (*evlist*) and friends event list (*fevlist*)

are taken and a union is performed on them. Then friendship value is calculated (*fval*) and that along with UEMAT value is considered for boosting using the equal weightage boosting technique.

```
nlist = union(evlist,fevlist);
nfl = size(nlist,2);
for l = 1 : nfl
val = 0;
count = 0;
        for k = 1:size(a,2)
                fval = frndsADJMAT(i,a(1,k));
                fval = (fval/size(a,2))* UEMAT(a(1,k),nlist(l));
                val = val + fval ;
        end
        ifisempty(find(nlist(1,l) == evlist(1,:),1))
                nUEMAT(i,nlist(l)) = val;
        else
                nUEMAT(i,nlist(l)) = (0.5 * UEMAT(i,nlist(1,l))) + (0.5 * val);
        end
        end
```

Algorithm 4.3 Boosting By Mutual Friends

4.5.2.2 Boosting By Preference Similarity

This takes into account similar interests and preferences by two users. If A and B have attended 3 events together and have responded in a similar manner to all three of them, then they have high level of trust. On the other hand if A and C have responded to 10 different events but only one of the events have rendered a similar response then it does not have a high level of trust.The algorithm is as follows: First both friendship value and user value are retrieved.*fsum* is the total number of events that are in common whereas *fval*is the friendship value that is similar to both users.The resultant value is divided by *fsum* .Again equal weightage technique is being used here for best results.

```
for k = 1:size(a,2)
fval = UEMAT(a(1,k),e);
        if fval == 0
                continue;
        end
        fval = as(1,k)*fval;
        val = val +  fval;
        fsum = fsum + as(1,k);
        end
        if fsum ~= 0
        ifisempty(find(e == evlist(1,:),1))
                nUEMAT(i,e) = val/fsum;
        else
```

*nUEMAT(i,e) = (0.5 * UEMAT(i,e)) + (0.5 * (val/fsum));*

end

end

Algorithm 4.4 Preference Similarity Boosting

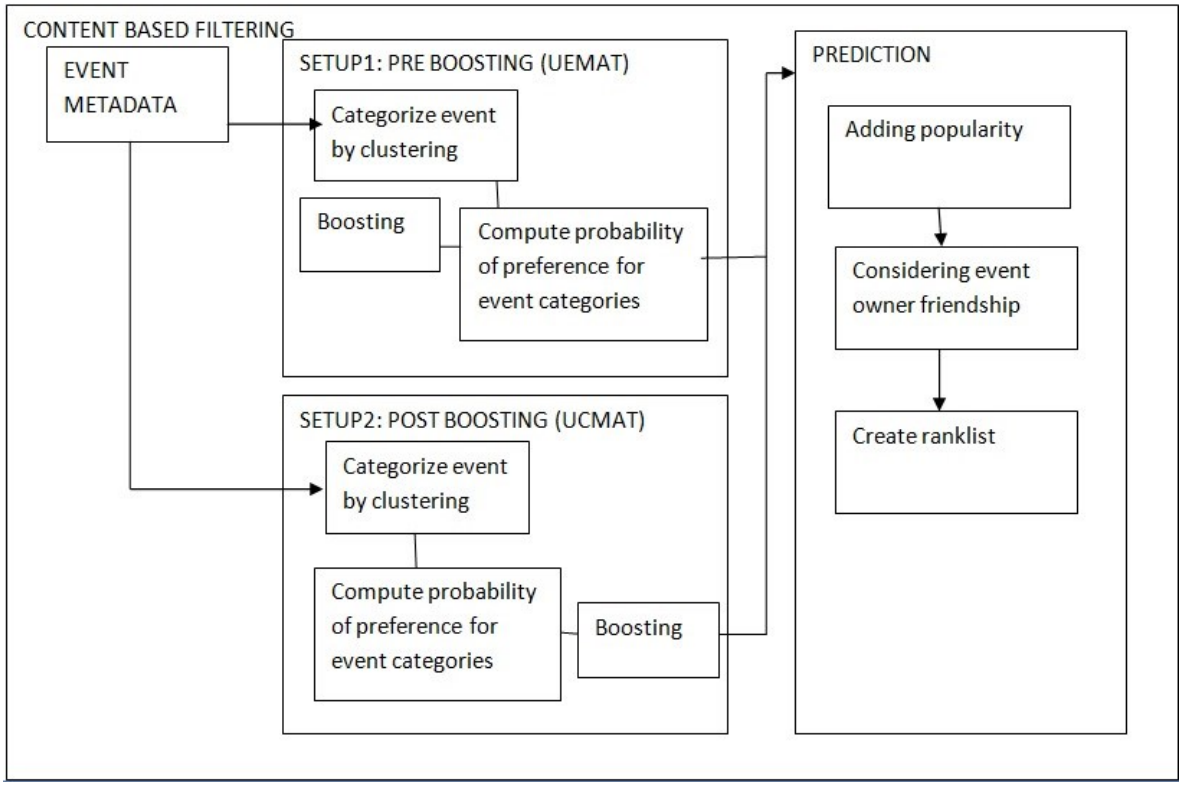

Figure 4.3 Data Analytics

4.6 CLUSTERING FOR CHAINS

The last phase is clustering for chains. It basically follows Lloyd's Algorithm. The steps are as follows:

Hyperspace Representation: A method for mapping chains to an *m*-dimensional (as opposed to *n*-dimensional)vector space. The mapping can be computed in time $O(nm)$. This method has a slightly differentmotivation than the one discussed above. Let *f* be the mapping from the set of all chains to R*m*andlet*d* be a distance function in R*m*. Furthermore, let p be a chain and denote by p*R*the reverse of p,that is, the chain that orders the same items as p, but in exactly the opposite way. The mapping *f*and distance *d* should satisfy

$d(f(\mathrm{p}), f(\mathrm{p}R)) = \max\{d(f(\mathrm{p}), f(\mathrm{p}0))\}$

$d(f(\mathrm{p}1), f(\mathrm{p}R1)) = d(f(\mathrm{p}2), f(\mathrm{p}R2)))$ for all p1 and p2.

Random Cluster Assignment: Randomly assign clusters before performing clustering and prediction.

Centroid computation: First initialize k centroids that are to be used in K Means on the data set X. Initialize the centroids to be random samples and randomly reorder the indices of samples. Next, find closest centroids which returns the closest centroids in idx for a dataset X where each row is a single example.idx = m x 1 vector of centroid assignments (i.e. each entry in range [1..K].Centroid computation for the cluster of chains is performed and the module returns the centroid computed as probabilistic order relationship matrices for each cluster which is then used for prediction. The algorithm is as

follows. Consider X - Dataset, where each row represents a Chain, K - number of Cluster,idx - Cluster assignments for the chains in X, probmat - Centroids of the cluster represented as probabilistic order relationship,n - number of chains, m - number of items.

```
[n,m] = size (X);
%Sets idx
idx = zeros(n, 1);
%probmat = zeros(m,m,K);

% INTIALIZATION
 %RANDOM INITIALIZATION OF CLUSTERS

        For     i = 1:n
                val = floor(rand() * 1000);
                idx(i,1) = mod(val,K) + 1;
        end

% CENTROID COMPUTATION - PROBABILISTIC ORDER RELATIONSHIPS
FOR CLUSTERS
probmat = computeCentroid(X,idx,K);
fprintf('\tInitialization DONE\n');
% END OF INITIALIZATION

%assnc - Number of cluster assignments in a single run of the algorithm
%Set assnc
assnc = 6;
z = 1;
% LOOP UNTIL ZERO RE-ASSIGNMENTS OF CLUSTERS
        while ( assnc> 0)
                fprintf('\t\ti:%d assn:%d\n',z,assnc);
                % RE-ASSIGN CLUSTERS TO CHAINS
                [idx,assnc] = findClosestCentroid(X,idx,probmat,K);
                  % CENTROID COMPUTATION - PROBABILISTIC ORDER
                RELATIONSHIPS FOR CLUSTERS
```

```
            probmat = computeCentroid(X,idx,K);
             z = z + 1;
        end
%END OF LOOP
End
```

Algorithm 4.5 Clustering For Chains

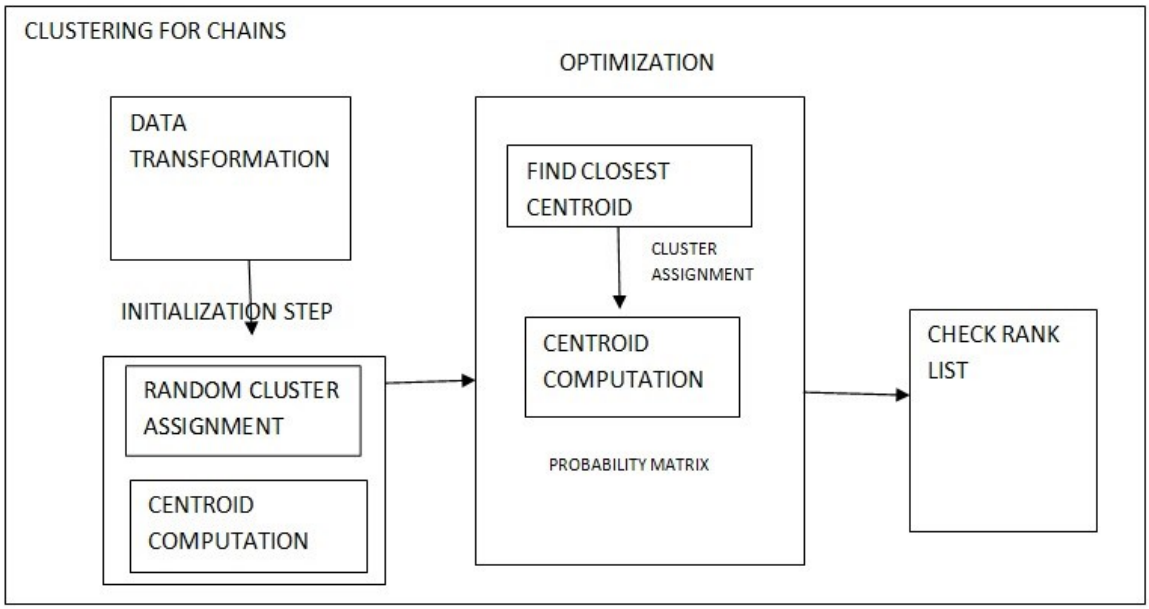

Figure 4.4 Clustering For Chains

4.7 SCREENSHOTS

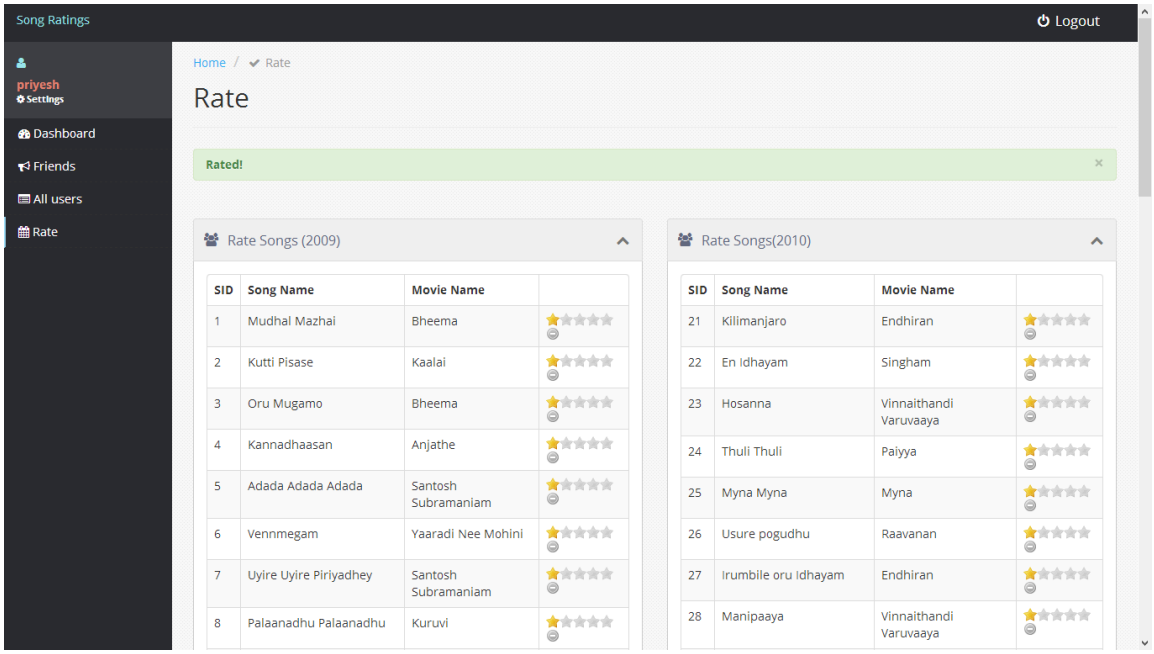

Figure 4.5 Rating

We have designed a music recommendation system. The database contains 100 songs of the years 2009, 2010, 2011, 2012 and 2013. We collected ratings from 150 users for the songs. Out of 150 we used 20% of the data as training set and 10% of the data as test set. The task was to predict ranking list for those users for the songs of 2013.

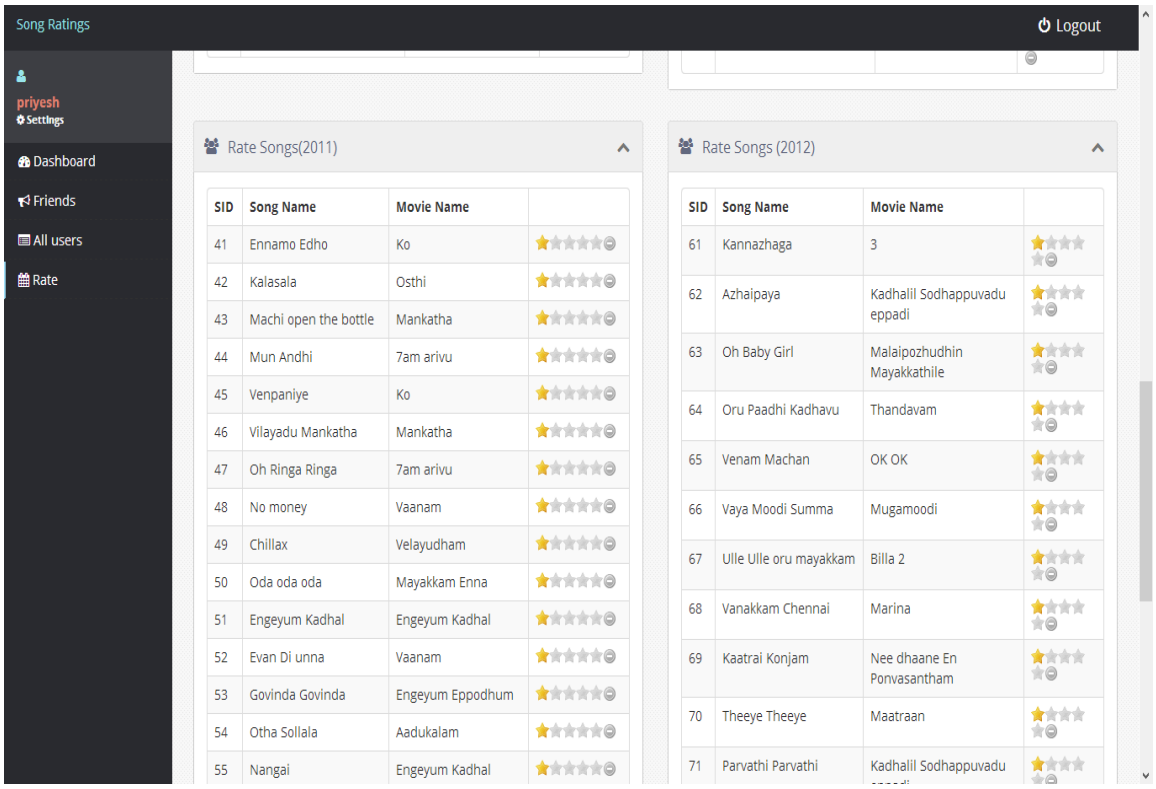

Figure 4.6 Ratings Of Users

4.8 PERFORMANCE EVALUATION

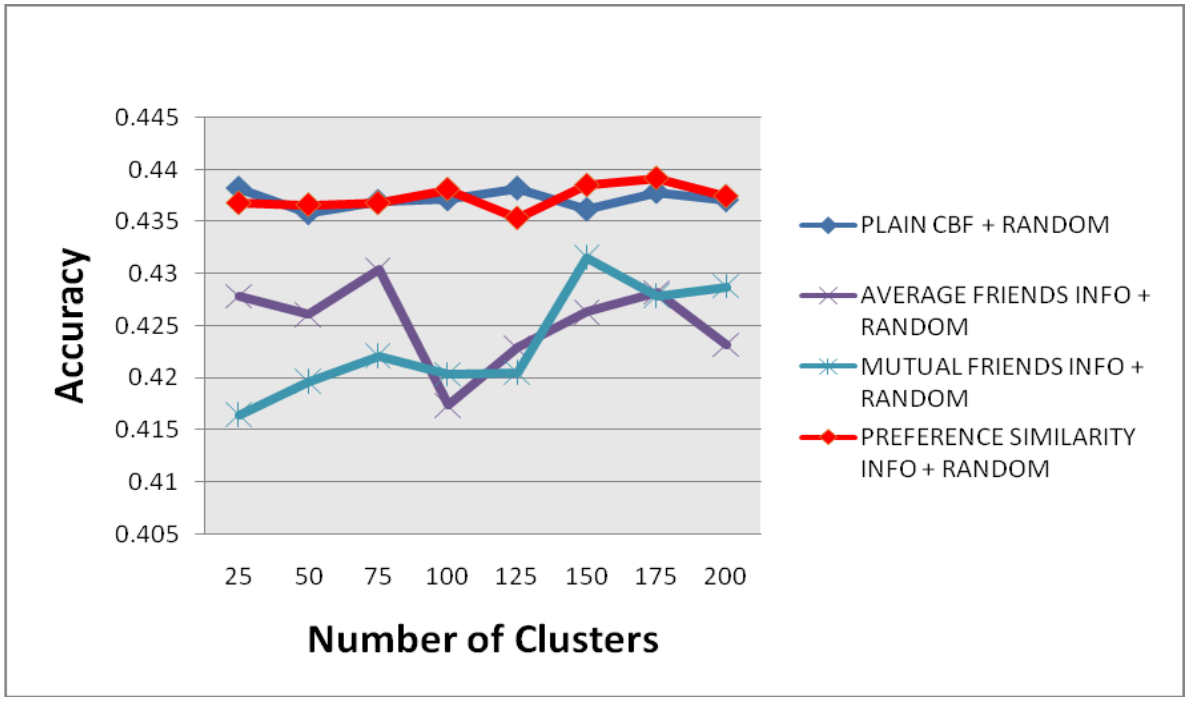

Figure 4.7 Old+New Users

Plain CBF + Random is considered as benchmark. We have shown the results for average friends information, mutual friends information, preference similarity information combined with random. CBF is done for 226 users and random for the remaining 360 users. On the whole, preference similarity information with random yields best results.

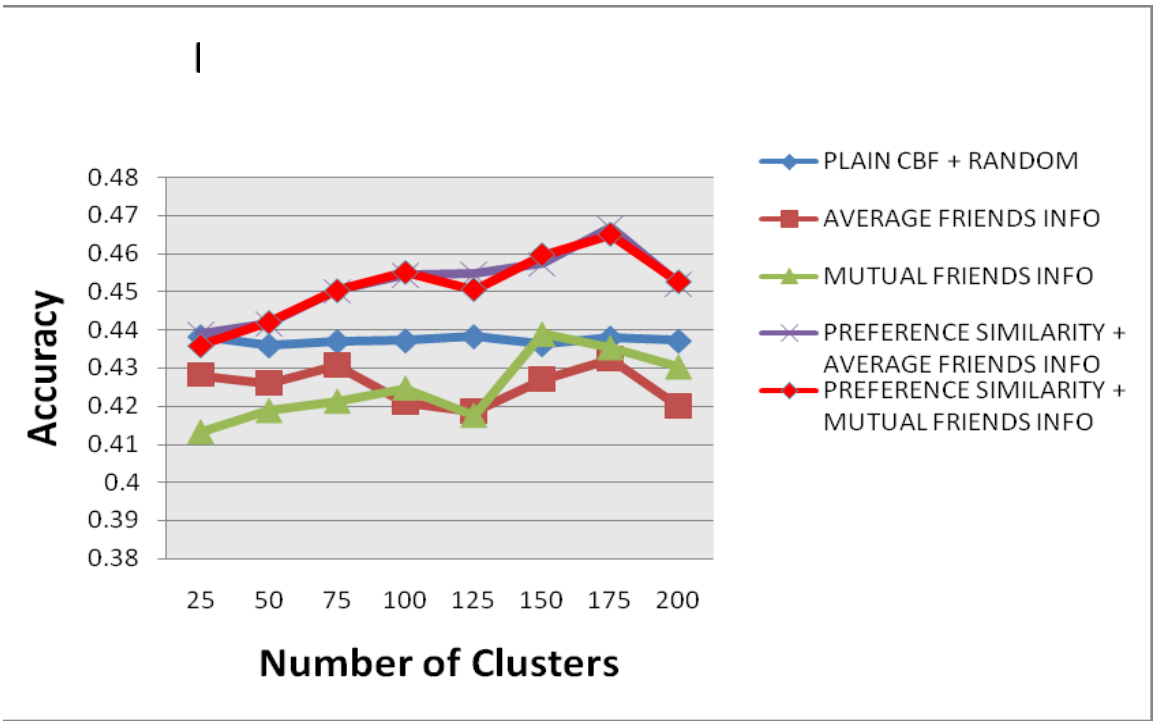

Figure 4.8 Popularity Boosting

Plain CBF + Random is considered as benchmark. We have shown the results for average friends information, mutual friends information, preference similarity information combined with average friends information. CBF is done for 226 users and random for the remaining 360 users. On the whole preference similarity information with average friends information yields best results.

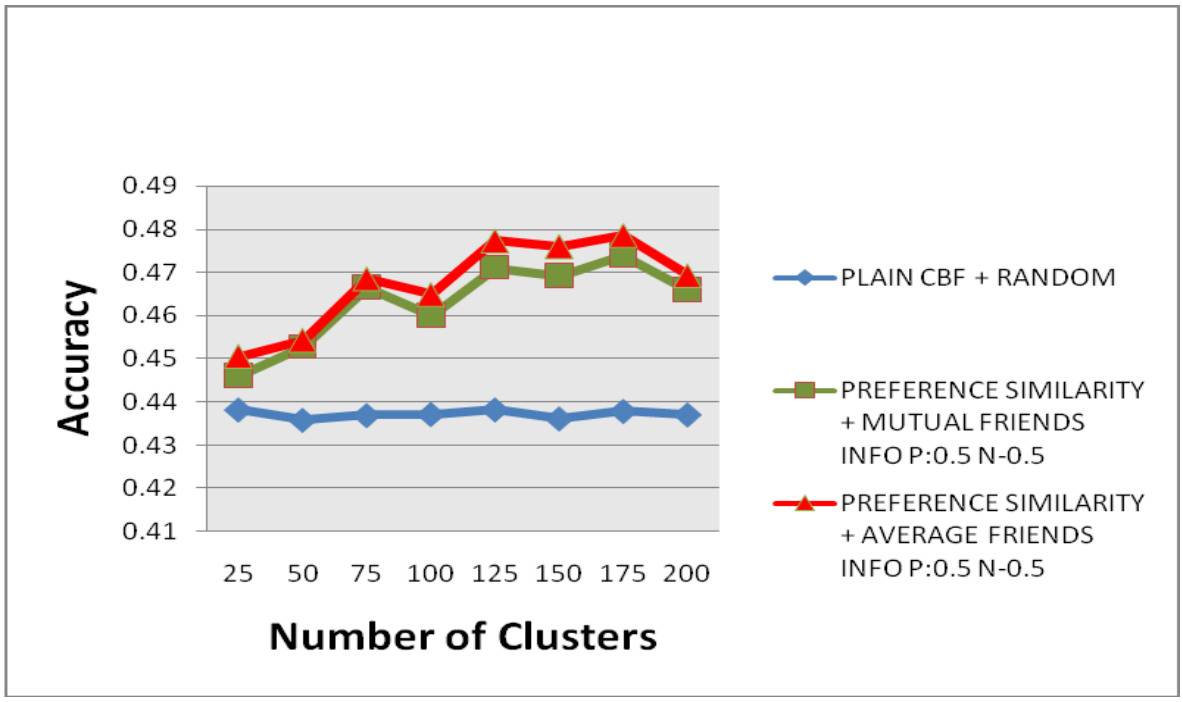

Figure 4.9 Preference Similarity

Here we boost all users and rank new events by popularity. Plain CBF with random is considered as the bench mark. Here we choose only preference similarity technique because it yielded better results than the other two methods. Also we take two different cases here, the first one with mutual friends and the next one with average friends. We also take into account positive and negative constant(0.5) values and from the above graph conclude that when taking only the preferred responses the model gives better results.

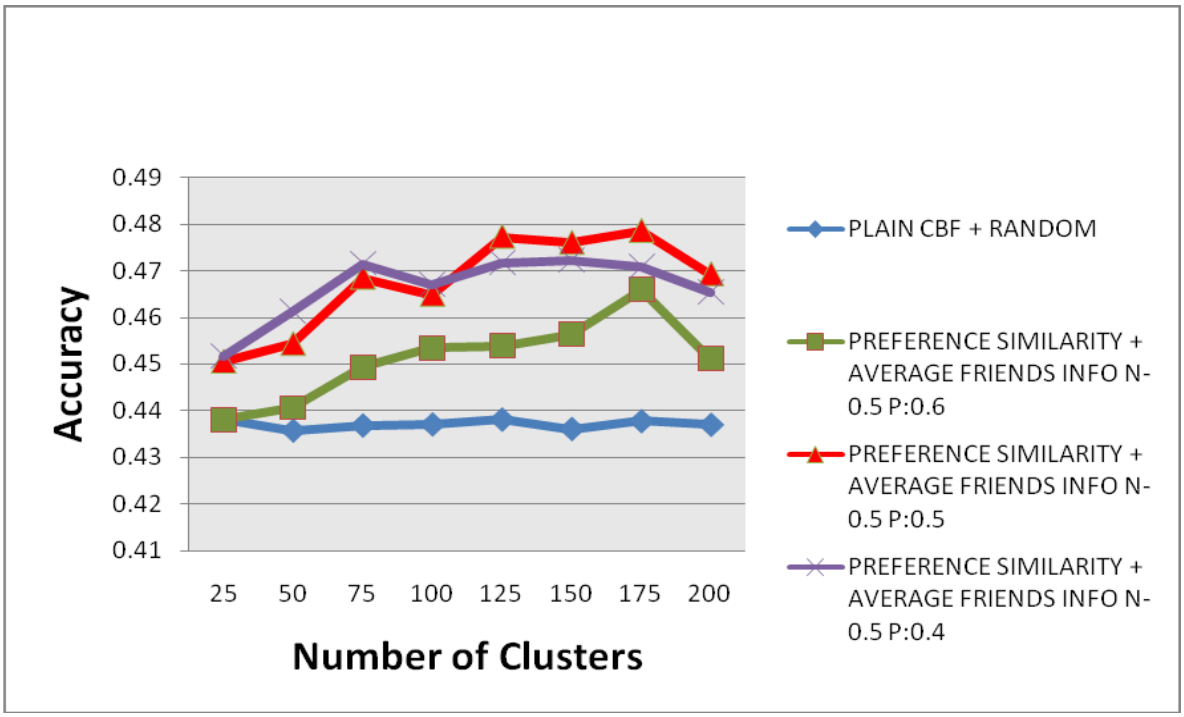

Figure 4.10 Popularity+Not Interested

Here we boost all users and rank new events by popularity. Plain CBF with random is considered as the bench mark. Here we choose only preference similarity technique because it yielded better results than the other two methods. Also we take three different cases here, all three with average friends information. We also take into account negative constant(0.5) values and vary P value from 0.4 to 0.6. From the above graph conclude that when taking only the 0.5 values we get best results.

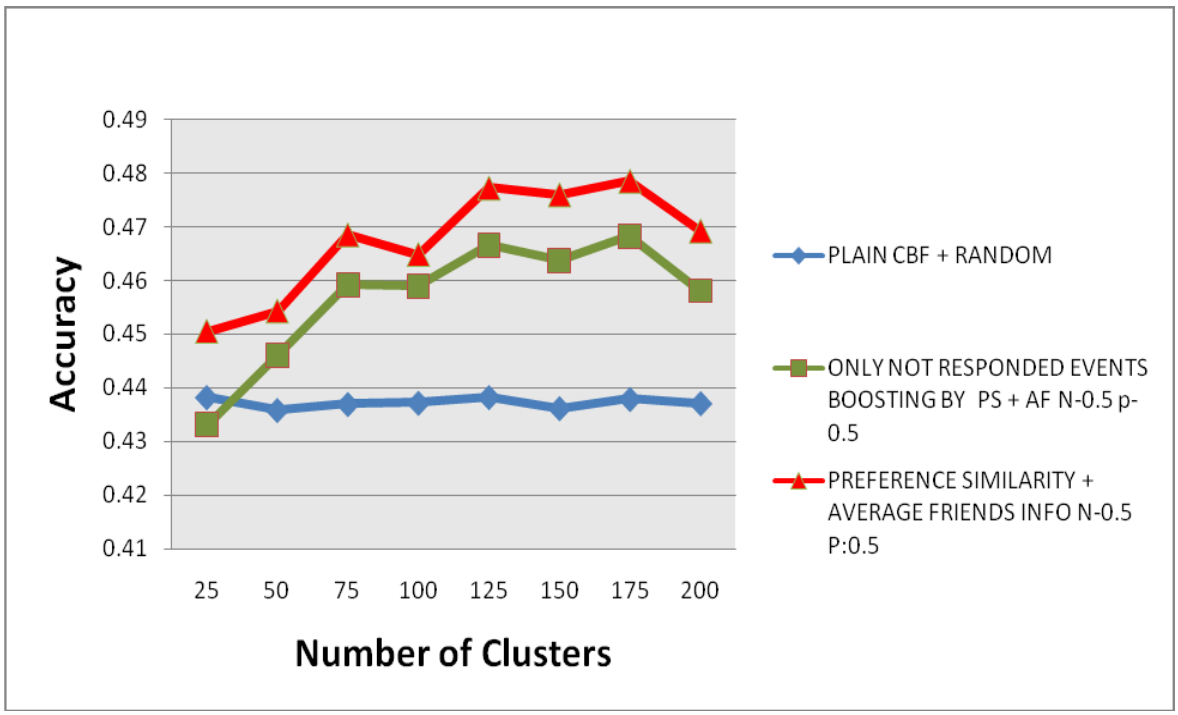

Figure 4.11 Complete And Partial Boosting

In the above case, Plain CBF with random is considered as benchmark. Partial boosting takes into account only not responded events and complete boosting takes both responded and not responded events into consideration. The P and N values of 0.5 are retained as detailed previously. Also preference similarity is used here and yield better results.

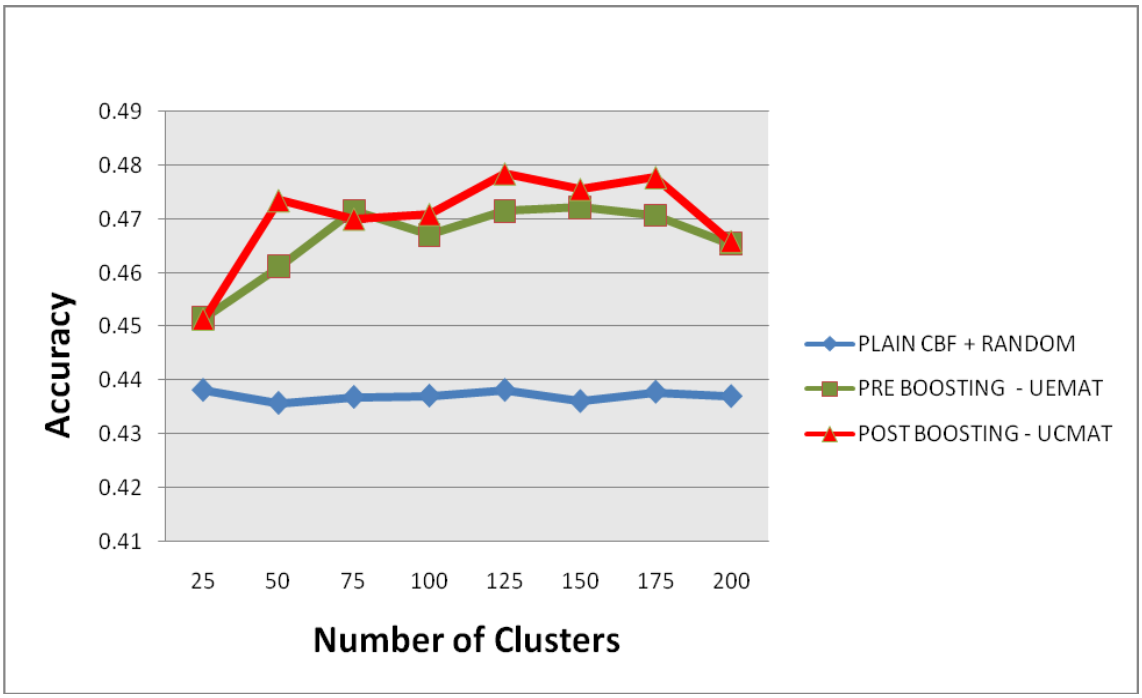

Figure 4.12 Pre Boosting and Post Boosting

Pre boosting refers to UEMAT whereas post boosting refers to UCMAT. Plain CBF and random is considered here as benchmark. Both pre boosting and post boosting yield good results but it is slightly better after clustering according to categories.

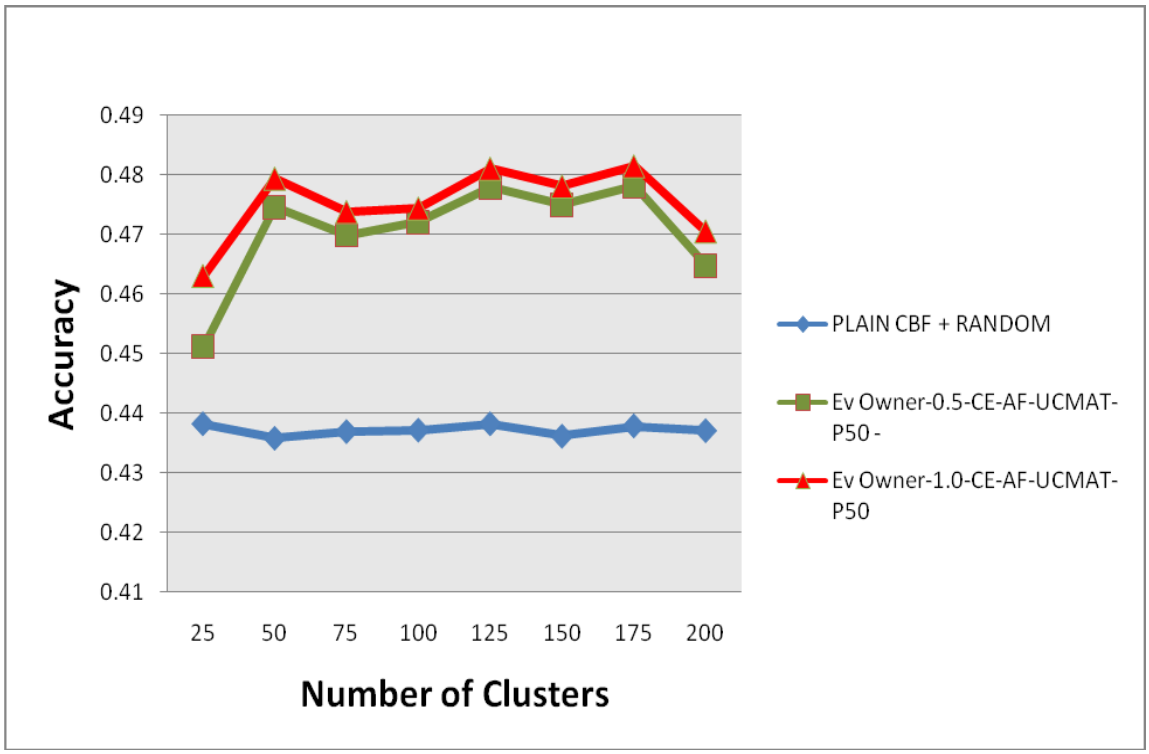

Figure 4.13 Adding Event Owner Friendship

If the owner of the event is in the users' friend circle then there are two possibilities. One is that the person will attend the event for sure(1.0) and the other is that there is a 0.5 probability. Both these results are much more efficient than the benchmark specified.

Table 4.2 Clustering of Chains and Boosting

	75	100	125	150	175	200	Accuracy
Plain CBF+Random	0.43692	0.43715	0.43819	0.43616	0.43785	0.43706	**0.43716**
Preference similarity+random	0.43678	0.43811	0.43534	0.4385	0.43918	0.43743	**0.43734**
Average friends information	0.43076	0.42096	0.41859	0.42709	0.43234	0.41983	**0.4254**
Preference similarity+average friends information	0.45014	0.45421	0.45466	0.4572	0.46678	0.45198	**0.45191**
Preference similarity+average friends information P 0.5 N -0.5	0.46853	0.46489	0.47729	0.47602	0.47856	0.46927	**0.46742**
Post boosting-UCMAT	0.47003	0.47088	0.47845	0.47554	0.47777	0.46585	**0.47042**
Event ownership	0.47373	0.47432	0.48102	0.47802	0.48141	0.47045	**0.47515**

The above table depicts the results for different boosting techniques with plain CBF as benchmark. As clearly seen, the techniques proposed by us yield better results. For example, the preference similarity plus average friends information is 3.79% higher than plain CBF plus random. It also varies for different clusters as shown above.

Table 4.3 Boosting All Users Plus Ranking Events By Popularity

MUTUAL FRIENDS INFO	0.41325	0.41881	0.4213	0.42469	0.4176	0.43884	0.43528	0.43025	0.4252
PREFERENCE SIMILARITY + MUTUAL FRIENDS INFO	0.43582	0.44195	0.45025	0.45503	0.45056	0.45966	0.465	0.4526	**0.45135**
PREFERENCE SIMILARITY + MUTUAL FRIENDS INFO P:0.5 N-0.5	0.4461	0.45282	0.46667	0.46008	0.47116	0.46921	0.47401	0.46596	0.46325
PREFERENCE SIMILARITY + AVERAGE FRIENDS INFO P:0.5 N-0.5	0.45054	0.45432	0.46853	0.46489	0.47729	0.47602	0.47856	0.46927	**0.46742**

Table 4.3 tells us about boosting all users plus ranking new events by popularity. As deciphered from above the result is highest on an average over all the clusters when we combine preference similarity with average friends information (increase by 4.22%).

Table 4.4 Popularity And Not Interested Events Position

POPULARITY AND NOT INTERESTED EVENTS POSITION									
PREFERENCE SIMILARITY + AVERAGE FRIENDS INFO N-0.5 P:0.6	0.43802	0.44062	0.44929	0.45336	0.45381	0.45636	0.46593	0.45113	0.451065
PREFERENCE SIMILARITY + AVERAGE FRIENDS INFO N-0.5 P:0.4	0.45161	0.46119	0.4715	0.46709	0.47158	0.47223	0.47076	0.46545	0.46642

In Table 4.4 we consider popularity and not interested events position with N constant at -0.5 and P varies from 0.4 to 0.6.

Table 4.5 Music Recommendation Results

TRAINING SET RESULTS			
	CLUSTERS		
MODELS OF CLUSTERING OF CHAINS (CC)	**10**	**15**	**20**
Conventional CC	0.46556	0.46526	0.47628
Unweighted Friends INFO boosted CC	0.56568	0.56728	0.57316
Mutual Friends INFO boosted CC	0.59558	0.600024	0.58948
Preference Similarity INFO boosted CC	**0.62552**	**0.60752**	**0.61198**
TEST SET RESULTS			
	CLUSTERS		
	10	15	20
Conventional CC	0.53174	0.52516	0.53904
Unweighted Friends INFO boosted CC	0.73878	0.70796	0.75538
Mutual Friends INFO boosted CC	0.7754	0.7848	0.76132
Preference Similarity INFO boosted CC	**0.8064**	**0.76513**	**0.7841**

The table above depicts the results achieved. In both the training and test data the results obtained from preference similarity information boosting is found to be the highest, around 27.466%.

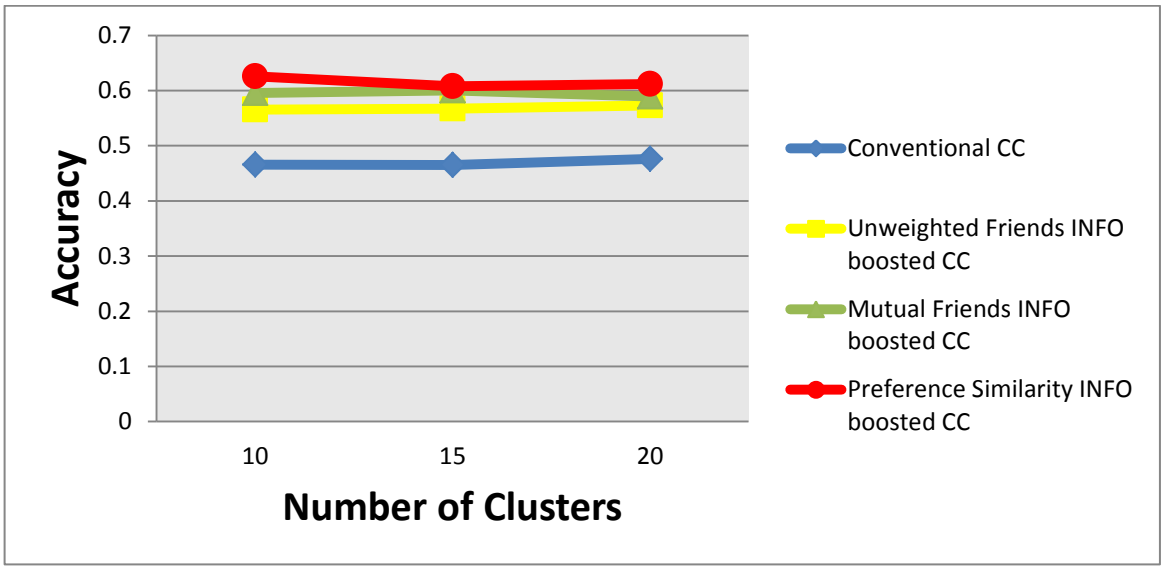

Figure 4.14 Training Results

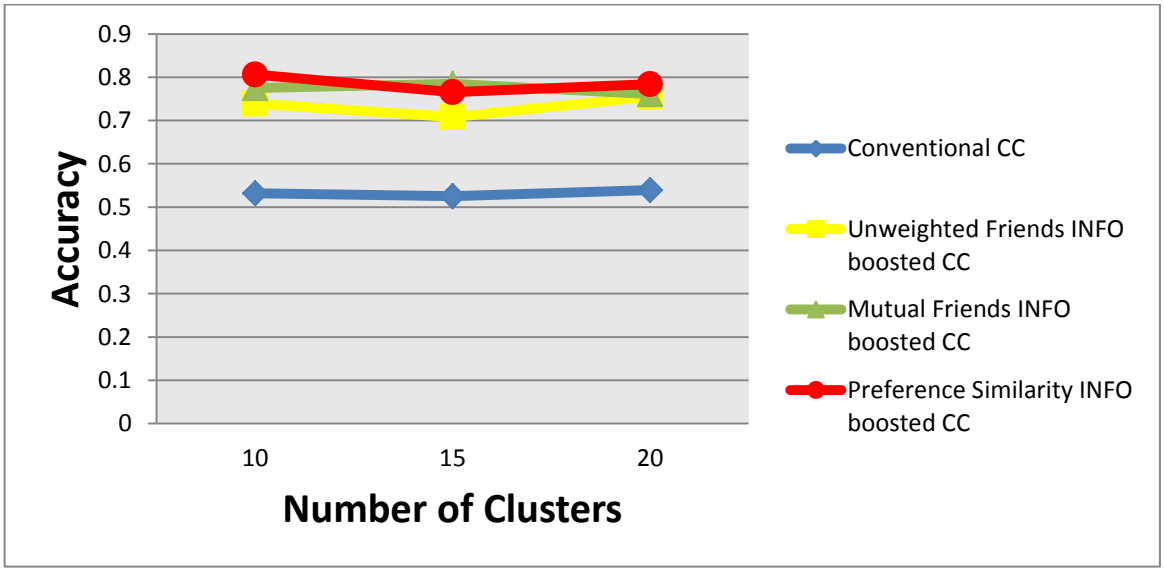

Figure 4.15 Test Results

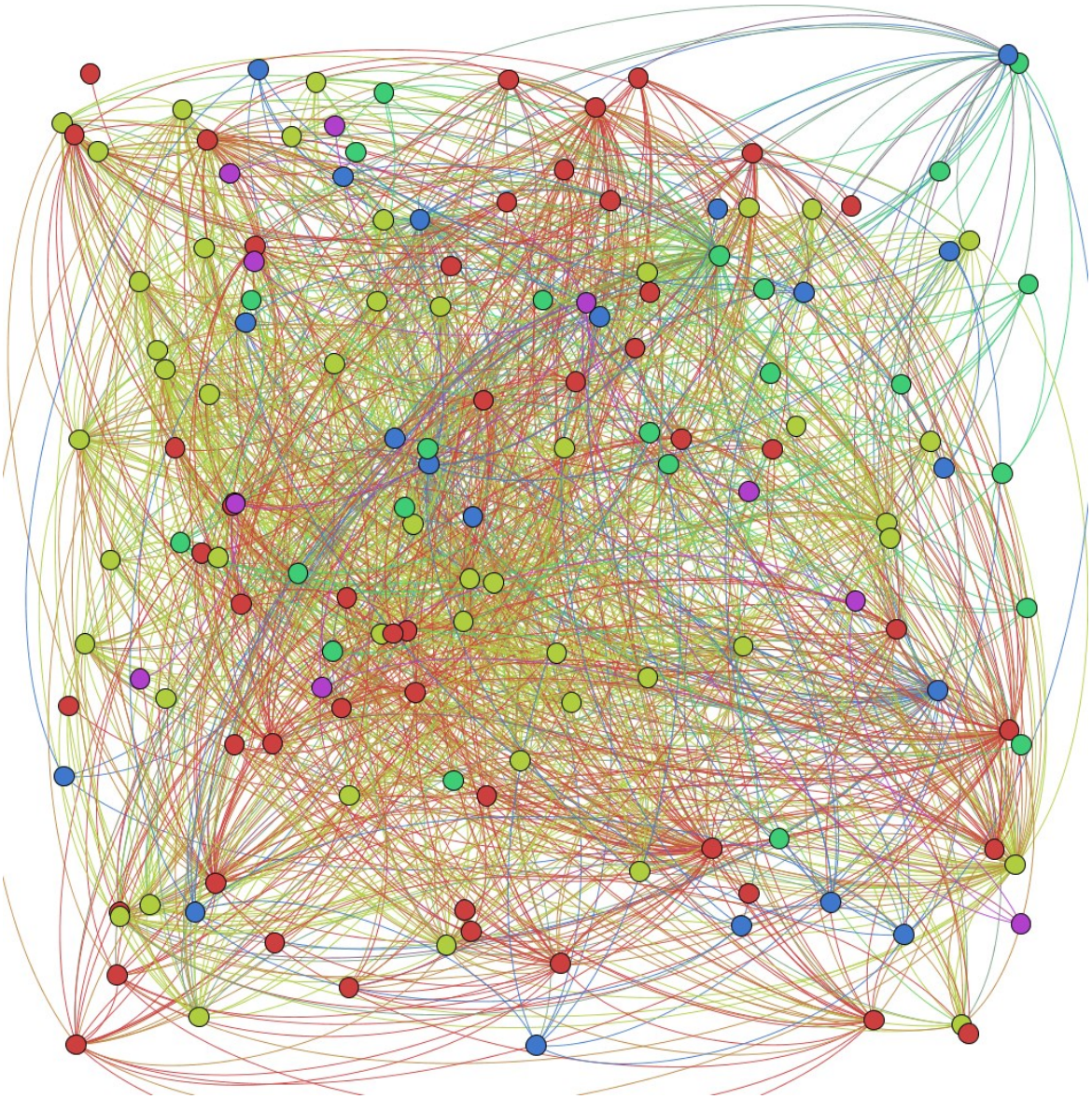

Figure 4.16 Friends Information - Music Recommender System

The above graph is created from the friend's information of the music recommender system. The graph contains 148 nodes and 1472 edges.

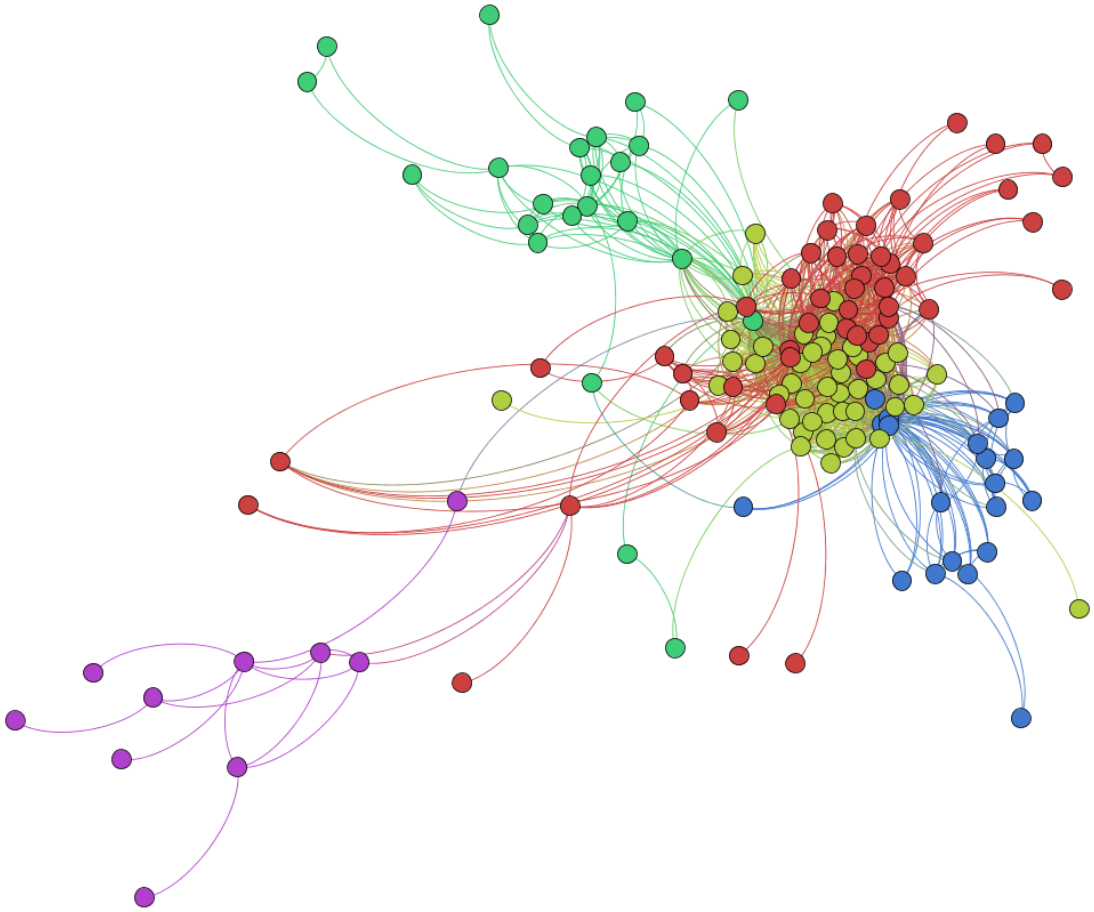

Figure 4.17 Community Results Using Gephi

The above graph is obtained as a result of running Community Detection algorithm on the friend's information available. As a result of which we have got five communities in the graph shown in five different colours respectively. It is evident from the formation of communities that, there is useful information in the available friend's information.

CHAPTER 5

REQUIREMENT ANALYSIS

5.1 HARDWARE REQUIREMENTS

1. Intel i5 processor or above
2. 6GB of RAM

5.2 SOFTWARE REQUIREMENTS

- OS - Windows XP or above
- Tools - MATLAB, PHP and GEPHI

5.2.1 MATLAB

MATLAB provides a range of numerical computation methods for analyzing data, developing algorithms, and creating models. The MATLAB language includes mathematical functions that support common

engineering and science operations. Core math functions use processor-optimized libraries to provide fast execution of vector and matrix calculations. MATLAB add-on products provide functions in specialized areas such as statistics, optimization, signal analysis, and machine learning. MATLAB provides tools to acquire, analyze, and visualize data, enabling you to gain insight into your data in a fraction of the time it would take using spreadsheets or traditional programming languages. You can also document and share your results through plots and reports or as published MATLAB code. MATLAB lets you access data from files, other applications, databases, and external devices. You can read data from popular file formats such as Microsoft Excel; text or binary files; image, sound, and video files; and scientific files such as netCDF and HDF. FilI/O functions let you work with data files in any format. Using MATLAB with add-on products, you can acquire data from hardware devices, such as your computer's serial port or sound card, as well as stream live, measured data directly into MATLAB for analysis and visualization. You can also communicate with instruments such as oscilloscopes, function generators, and signal analyzers. MATLAB lets you manage, filter, and preprocess your data. You can perform exploratory data analysis to uncover trends, test assumptions, and build descriptive models. MATLAB provides functions for filtering and smoothing, interpolation, convolution, and fast Fourier transforms (FFTs). MATLAB provides built-in 2-D and 3-D plotting functions, as well as volume visualization functions. You can use these functions to visualize and understand data and communicate results. Plots can be customized either interactively or programmatically.

5.2.1.1 Documenting and Sharing Results

You can share results as plots or complete reports. MATLAB plots can be customized to meet publication specifications and saved to common graphical and data file formats. You can automatically generate a report when you execute a MATLAB program. The report contains your code, comments, and program results, including plots. Reports can be published in a variety of formats, such as HTML, PDF, Word, or LaTeX.

5.2.2 PHP

PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language. PHP is now installed on more than 244 million websites and 2.1 million web servers. Originally created by Rasmus Lerdorf in 1995, the reference implementation of PHP is now produced by The PHP Group. While PHP originally stood for Personal Home Page, it is now said to stand for PHP: Hypertext Preprocessor, a recursive acronym.

The PHP language was originally implemented as an interpreter, and this is still the most popular implementation. Several compilers have been developed which decouple the PHP language from the interpreter. Advantages of compilation include better execution speed, static analysis, and improved interoperability with code written in other languages. PHP compilers of note

include Phalanger, which compiles PHP into Common Intermediate Language (CIL) bytecode, and HipHop, developed at Facebook and now available as open source, which transforms the PHP Script into C++, then compiles it, reducing server load up to 50%. PHP source code is compiled on-the-fly to an internal format that can be executed by the PHP engine. In order to speed up execution time and not have to compile the PHP source code every time the web page is accessed, PHP scripts can also be deployed in executable format using a PHP compiler. Code optimizers aim to enhance the performance of the compiled code by reducing its size , merging redundant instructions and making other changes that can reduce the execution time. With PHP, there are often opportunities for code optimization. An example of a code optimizer is the eAccelerator PHP extension.

5.2.3 GEPHI

Gephi is an open-source network analysis and visualization software package written in Java on the NetBeans platform. Gephi has been selected for the Google Summer of Code in 2009, 2010, 2011, and 2012.Gephi has been used in a number of research projects in the university, journalism and elsewhere, for instance in visualizing the global connectivity of New York Times content[3] and examining Twitter network traffic during social unrest[4][5] along with more traditional network analysis topics. The Gephi Consortium is a French non-profit corporation which supports development of future releases of Gephi. Members include SciencesPo, Linkfluence, WebAtlas, and Quid.

CHAPTER 6

CONCLUSION AND FUTURE WORK

Traditionally, collaborative filtering systems have relied heavily on similarities between the ratings of users as a way to differentially rate the prediction contributions of different users. In this paper we have argued that similarity on its own may not be sufficient, that other factors might also have an important role to play. Specifically we have introduced the notion of trust in reference to the degree to which one might trust a specific user when it comes to making a specific rating prediction. In this paper we argue that incorporating trust and popularity into recommender systems can lead to an increase in accuracy and quality of recommendations. Based on this idea, we proposed a trust model which implicitly quantifies the degree of trust a user holds for another user, and proposed new trust-based recommendation strategies which incorporate the new model into the standard collaborative filtering recommender system. For a baseline comparison, we used the traditional similarity-based collaborative filtering strategy. The empirical results indicate that trust, popularity are very effective tools for improving recommender systems around 28%.

REFERENCES

[1] Aimeur F.S.M, Onana. (2006) “Better Control on Recommender Systems”. Proceedings of the 8th IEEE International Conference on E-Commerce Technology.

[2] Akaho S, Kamishima T. (2009)“Mining Complex Data, volume 165 of Studies in Computational Intelligence”.Chapter Efficient Clustering for Orders, pp 261–279.

[3] Antti, Ukkonen. (2011) “Clustering algorithms for chains”. Journal of Machine Learning Research 12 ,pp.1389-1423.

[4] Avesani P, Massa P. (2007). “Trust-aware Recommender Systems”. RecSys’07.

[5] Breese J.S, Heckerman, Kadie. (1998) “Empirical analysis of predictive algorithms for collaborative filtering”.14th Conf. on Uncertainty in Artificial Intelligence”,pp. 43–52.

[6] Golbeck J, Hendler J. (2005) “Accuracy of metrics for inferring trust and reputation insemantic web-based social networks”. In Proceedings of EKAW’04, LNAI 2416, pp.278.

[7] He J, Konstan J, Terveen L G, Riedl. (2005) “Evaluating collaborative filtering recommender systems.” ACM Trans. Inf. Syst. 22 pp. 5–53.

[8] Martin D, Murphy T.B. (2003) “Mixtures of distance-base models for ranking data”. Computational Statistics & Data Analysis, 41:pp.645–655.

[9] O'Donovan. J, Smyth.B. (2005) “Trust in Recommender Systems”. Proceedings of the 10th international conference on Intelligent user interfaces.